សាកលវិទ្យាល័យភូមិន្ទនីតិសាស្ត្រនិងវិទ្យាសាស្ត្រសេដ្ឋកិច្ច

Université Royale de Droit et des Sciences Économiques

Royal university of Law and Economics

សារណាបញ្ចប់ការសិក្សា

ការសិក្សាប្រៀបធៀបនៃភាសាសរសេរកម្មវិធី

Java និង C#

ស្រាវជ្រាវពីថ្ងៃទី ៨ ខែ មេសា ឆ្នាំ ២០២៤ ដល់ថ្ងៃទី ១០ ខែ សីហា ឆ្នាំ ២០២៤

ស្រាវជ្រាវដោយ

និស្សិតឈ្មោះ **កញ្ញា លុយ ជាណេ**

កញ្ញា ឃួច សុភ័ក្រកញ្ញាកេត

ណែនាំដោយ

សាស្ត្រាចារ្យជំនួយ បណ្ឌិត **ឥន ហេង**

ថ្នាក់បរិញ្ញាបត្រ សេដ្ឋកិច្ចព័ត៌មានវិទ្យា

ជំនាន់ទី ១៩

ឆ្នាំចូលសិក្សា ២០២០

ឆ្នាំសរសេរសារណា ២០២៤

សេចក្តីថ្លែងអំណរគុណ

នាងខ្ញុំឈ្មោះ **លុយ ជាណេ** និង **ឃួង សុភ័ក្ត្រកញ្ញេត** ជានិស្សិតឆ្នាំទី៤ នៃដេប៉ាតឺម៉ង់សេដ្ឋកិច្ច ព័ត៌មាន វិទ្យានៅសាកលវិទ្យាល័យភូមិន្ទនីតិសាស្ត្រ និងវិទ្យាសាស្ត្រសេដ្ឋកិច្ចជំនាន់ទី១៩ នៅក្នុងឆ្នាំសិក្សា ២០២៣- ២០២៤។ យើងខ្ញុំទាំងពីរនាក់សូមធ្វើការអរគុណ ដឹងគុណកតញ្ញូ និងសេចក្តីគោរពជូនដល់លោកអ្នក មានគុណទាំងឡាយមាន៖

លោកឪពុក អ្នកម្តាយដែលបានផ្តល់កំណើតឱ្យកូនៗ បានរស់រានមានជីវិត និងខិតខំចិញ្ចឹមបីបាច់ថែ រក្សា មើលការខុសត្រូវណែនាំទំនុកបម្រុងផ្តល់ឱ្យនូវអ្វីដែលចាំបាច់ក្នុងការអប់រំកូនឱ្យក្លាយទៅជាមនុស្សម្នាក់ ដែល មានសក្តានុពល អំណត់អត់ធ្មត់នៅក្នុងគ្រួសារ និងសង្គមជាតិទាំងមូលផងដែរ។

ឯកឧត្តមបណ្ឌិតសាកលវិទ្យាធិការ សាកលវិទ្យាធិការរង ព្រឹទ្ធបុរស ព្រឹទ្ធបុរសរង នៃសាកលវិទ្យាល័យ ភូមិន្ទនីតិសាស្ត្រ និងវិទ្យាសាស្ត្រសេដ្ឋកិច្ចដែលតែងតែស្វែងរកនូវភាពងាយស្រួលដល់ការសិក្សាជូនដល់និស្សិត គ្រប់រូបរួមទាំងខ្ញុំទាំងពីរនាក់ ដើម្បីឱ្យការសិក្សាមានភាពរលូនទៅមុខប្រកបដោយប្រសិទ្ធភាព នាកំឡុងពេល សិក្សាថ្នាក់បរិញ្ញាបត្រសេដ្ឋកិច្ចព័ត៌មានវិទ្យានេះ។

លោក លោកស្រីបុគ្គលិកសិក្សា និងលោកគ្រូ អ្នកគ្រូសាស្ត្រាចារ្យទាំងអស់ដែលតែងតែ ខិតខំបង្ហាត់ បង្រៀន ស្រាវជ្រាវនូវ មេរៀនថ្មីៗ ចែករំលែករាល់បទពិសោធន៍ផ្សេងៗ ដល់និស្សិតដោយក្តីអាណិតស្រឡាញ់ ចេញពីបេះដូងចាប់តាំងពីថ្នាក់ឆ្នាំមូលដ្ឋាន ដល់បញ្ចប់ថ្នាក់បរិញ្ញាបត្រនេះ។

សាស្ត្រាចារ្យជំនួយបណ្ឌិត **ឯិន ហោច** ដែលបានណែនាំក្នុងការសរសេរសារណាបញ្ចប់ថ្នាក់ បរិញ្ញាបត្រ សេដ្ឋកិច្ចព័ត៌មានវិទ្យានេះ។ លោកបានលះបង់នូវពេលវេលាដ៏មានតម្លៃរបស់លោកមកបង្ហាត់ បង្រៀនស្វះស្វែង រកនូវឯកសារពាក់ព័ន្ធនានាត្រួតពិនិត្យកែសម្រួលពិភាក្សាផ្តល់នូវដំបូន្មាន ឬគំនិតក្នុងការសរ សេរសារណាបញ្ចប់ ថ្នាក់បរិញ្ញាបត្រនេះ។ ដោយមានការជួយជ្រោមជ្រែងពីសំណាក់លោកសាស្ត្រាចារ្យអស់ពី ចិត្ត ទើបធ្វើឱ្យ សារណានេះបញ្ចប់ដោយជោគជ័យជាស្តាពរយ៉ាងរលូន។

ជាចុងក្រោយ យើងខ្ញុំទាំងពីររូបសូមគោរពជូនពរដល់អ្នកមានគុណទាំងពីរ ឯកឧត្តមបណ្ឌិតសាកល វិទ្យាធិការ សាកលវិទ្យាធិការរង ព្រឹទ្ធបុរស ព្រឹទ្ធបុរសរង សាស្ត្រចារ្យគ្រប់មុខវិជ្ជាទាំងអស់ ព្រមទាំងនិស្សិតគ្រប់ រូប សូមទទួលបាននូវពុទ្ធពរទាំង ៤ប្រការគឺ អាយុ វណ្ណៈ សុខៈ ពលៈ កុំបីឃ្លៀងឃ្លាតឡើយ ព្រមទាំងចៀស ចាក ផុត ពីជំងឺទាំងពួង។

អារម្ភកថា

នៅក្នុងយុគសម័យឌីជីថលនេះ បច្ចេកវិទ្យាត្រូវបានគ្របដណ្តប់លើគ្រប់វិស័យទៅហើយមិនថាកសិកម្ម ឧស្សាហកម្ម ទូរគមនាគមន៍ ប្រព័ន្ធផ្សព្វផ្សាយ ការកំសាន្ត ពាណិជ្ជកម្ម វិស័យអប់រំ។ល។ វាបានជួយសម្រួលដល់ការងាររបស់មនុស្សជាតិបានមួយផ្នែកធំរួមទាំងជីវភាពរស់នៅផងដែរ។ កំណើននៃការកើនឡើងចំនួនមនុស្សនៅលើពិភពលោកវាជាបញ្ហាមួយនៃការកើតឡើងនូវតម្រូវការជាច្រើនឥតឈប់ឈរស្របជាមួយ និងការគាំទ្រនៅបញ្ហានេះទើបអ្នកស្រាវជ្រាវជាច្រើនរូបបានខិតខំបង្កើតនូវគំនិតប្រឌិតថ្មីៗជាច្រើនទៅលើបច្ចេកវិទ្យាដើម្បីបំពេញសេចក្តីត្រូវការ របស់មនុស្សជាតិនៅលើពិភពលោកនេះដែលវាជាហេតុធ្វើឱ្យបច្ចេកវិទ្យាកាន់តែវិវត្តទំនើបទៅៗ ពីមួយដំណាក់កាលទៅមួយដំណាក់កាលយ៉ាងឆាប់រហ័សទើបធ្វើឱ្យមានការនិយាយតៗគ្នាថា បច្ចេកវិទ្យានៅសម័យសតវត្សន៍ទី២១នេះ បានកែប្រែរបៀបរស់នៅរបស់មនុស្សជាតិស្ទើរតែទាំងស្រុង។

ជាមួយនឹងការអភិវឌ្ឍន៍យ៉ាងឆាប់រហ័សនៃឧស្សាហកម្មកម្មវិធី មនុស្សកាន់តែច្រើនចង់រៀនភាសាសរសេរកម្មវិធី។ ប៉ុន្តែសព្វថ្ងៃនេះមានភាសាសរសេរកម្មវិធីដែលអាចប្រើបានមានតែមួយចំនួនប៉ុណ្ណោះ ដែលយើងអាចអនុវត្តបានយ៉ាងទូលំទូលាយ។ ដូចនេះហើយទើបពួកយើងទាំងពីរនាក់និងលើកយកភាសាសរសេរកម្មវិធីពេញនិយមបំផុតចំនួនពីរគឺ C# និង Java មកធ្វើការប្រៀបធៀបជាមួយគ្នា ដើម្បីជាជំនួយដល់អ្នកដែរចង់ស្វែងយល់ពីភាសាទាំងនេះអោយកាន់តែច្បាស់។ហើយពួកខ្ញុំទាំងពីរនាក់សង្ឃឹមថាការសិក្សាស្រាវជ្រាវនូវប្រធានបទមួយនេះ បានចូលរួមចំណែកក្នុងការអភិវឌ្ឍន៍ ទៅលើវិស័យបច្ចេកវិទ្យាបានមួយកម្រិតផងដែរជាពិសេសលើផ្នែកភាសាសរសេកម្មវិធីនេះ។

រាល់សកម្មភាព ឬការសិក្សាស្រាវជ្រាវមួយដែលមានលទ្ធផលល្អប្រកបដោយប្រសិទ្ធភាពទៀតនោះតែងតែមាននូវរាល់ឧបសគ្គឆ្លងកាត់ជាច្រើន។ ដូចគ្នានេះដែរការយកប្រធានបទនេះមកធ្វើការស្រាវជ្រាវចងក្រងធ្វើជា សារណារ ពួកយើងទាំងពីរនាក់បានជួបបញ្ហាជាច្រើន ដូចជាការខ្វះខាតនូវឯកសារ ចំណេះដឹងមួយចំនួន ។ល។ ជាសំណាងល្អដោយទទួលបានការណែនាំពីសាស្ត្រាចារ្យជំនួយបណ្ឌិត **ឆិន ហោច** និងការស្វះស្វែងរកនូវទិន្នន័យពាក់ព័ន្ធនានាតាមប្រព័ន្ធអ៊ីនធឺណេត (Internet) ទើបធ្វើឱ្យសារណារមួយនេះបញ្ចប់ដោយ ជោគជ័យ។ ទោះបីពួកយើងទាំងពីរនាក់ខិតខំអស់ពីសមត្ថភាពយ៉ាងណាក្តីក៏ការសិក្សាស្រាវជ្រាវមួយនេះនៅ មានចន្លោះខ្វះខាត បញ្ហាបន្តិចបន្តួចជាយថាហេតុ អាស្រ័យហេតុនេះពួកខ្ញុំទាំងពីរអ្នកសង្ឃឹមថាអ្នកអាននឹងអធ្យាស្រ័យអត់ឱនឱ្យពួកយើងទាំងពីរនាក់ រាល់កំហុសដែលកើតមានឡើង ទាំងចេតនា នឹងអចេតនាដោយក្តីអនុគ្រោះ។

មាតិកា

ទំព័រ

ជំពូកទី១

រំលឹកទ្រឹស្តី

ជំពូកទី២

ប្រៀបធៀប

ជំពូកទី៣

ការអនុវត្តនិងប្រសិទ្ធភាព

សេចក្តីសន្និដ្ឋាន និងការផ្តល់អនុសាសន៍

បញ្ជីអក្សរកាត់

C# = C-Sharp

LINQ = Language Integrate

SDK = Software Development Kit

API = Application Programming Interface

5GL = Fifth-generation Languages

JFC = Java Foundation Classes

.NET = National Eligibility Test

IDE = Integrated Development Environment

JVM = Java Virtual Machine

CLR = Common Languages Runtime

I/O = Input / Output

បញ្ជីតារាង

បញ្ជីរូបភាព

បញ្ជីដ្យាក្រាម

សេចក្តីផ្តើម

១. ចំណាំបញ្ហានៃការស្រាវជ្រាវ

នៅក្នុងពេលបច្ចុប្បន្ននេះ វិស័យបច្ចេកវិទ្យាមានភាពរីកចម្រើនយ៉ាងខ្លាំង ពីមួយថ្ងៃទៅមួយថ្ងៃ ហើយគេបានយកវាទៅប្រើប្រាស់ ពាសពេញសកលលោក។ មិនត្រឹមតែប៉ុណ្ណោះវិស័យ ព័ត៌មានវិទ្យាក៏បានដើរតួរនាទីយ៉ាងសំខាន់ក្នុងការអភិវឌ្ឍសេដ្ឋកិច្ចរបស់ប្រទេសជាតិ។ លើសពីនេះទៅទៀតគេបានចាត់ទុកវិស័យនេះជាវិស័យចំបងក្នុងការជម្រុញអជីវកម្ម និងសេវាកម្ម ឲ្យកាន់តែរីកចម្រើនផងដែរ។ ជាក់ស្តែងបើយើងក្រឡេកទៅមើលបណ្តាប្រទេសដែលមានការរីក ចម្រើនខ្លាំងដូចជាប្រទេស សហរដ្ឋអាមេរិច រុស្សី ជប៉ុន កូរ៉េ ចិន និងប្រទេសដែលនៅសមាគមន៍ អឺរ៉ុបជាដើមបានយកវិស័យព័ត៌មានវិទ្យា កំណត់ថាជាវិស័យសំខាន់ក្នុងការសម្រួលកិច្ចការដឹកនាំប្រទេសជាតិរបស់ខ្លួន ។

ដោយឡែកយើងក្រឡេកមកមើលប្រទេសកម្ពុជាយើងវិញ វិស័យព័ត៌មានវិទ្យាក៏កំពុងតែ មានការរីកចម្រើនផងដែរ ដែលធ្វើឲ្យមានការអភិឌ្ឍយ៉ាងឆាប់រហ័ស ទាំងផ្នែកសេដ្ឋកិច្ច សង្គមកិច្ច និងស្ទើតែគ្រប់វិស័យ។ បើយើងក្រលែកមកមើលទៅលើភាសាសរសេរកម្មវិធី និងបច្ចេកវិទ្យា តែងតែបន្តការអភិវឌ្ឍន៍យ៉ាងសកម្ម។ មានភាសាសរសេកម្មវិធីជាច្រើនត្រូវបានបង្កើតឡើងហើយត្រូវបានប្រើប្រាស់យ៉ាងទូលំទូលាយសម្រាប់ការអភិវឌ្ឍកម្មវិធីពហុកម្មវិធី។ ដូចនេះផងដែរយើងនឹងធ្វើការសិក្សាស្វែងយល់ពីភាពខុសគ្នារវាងភាសាពីរ ដែរអាចនាំយើងមកនូវការជ្រើសរើសភាសាមួយដែរល្អជាង សម្រាប់ការប្រើប្រាស់ជាមួយនិងតម្រូវការរបស់យើង។ ដោយហេតុនេះហើយបានជាក្រុមរបស់យើងខ្ញុំសម្រេចចិត្តជ្រើសរើសនៅប្រធានបទមួយមកធ្វើការស្រាវជ្រាវនោះគឺ **ការសិក្សាប្រៀបធៀបនៃភាសាសរសេរកម្មវិធី**Java**និង**C# ដើម្បីផ្តល់នៅភាពងាយស្រួលដល់សិស្សានុសិស្សឬអ្នកដែរចង់បន្ថែមនៅចំណេះដឹង ក្នុងការសិក្សាស្វែងយល់ពីភាសាសរសេកម្មវិធីទាំងនេះ។

២. ចំណោទបញ្ហានៃការស្រាវជ្រាវ

ភាសាសរសេរកម្មវិធីគឺមានចំនួនយ៉ាងច្រើននៅក្នុងពេលបច្ចុប្បន្ននេះ មនុស្សតែងតែមានការពិបាកអំពីការជ្រើសរើសភាសាសរសេរកម្មវិធីណាមួយដែលគួររៀន និងគួរប្រើដើម្បីបង្កើតកម្មវិធីជាក់លាក់មួយ។ភាសាសរសេរកម្មវិធី C# និង Java គឺជាភាសាដែលទទួលបានការពេញនិយមបំផុតនៅក្នុងពិភពនៃការអភិវឌ្ឍន៍កម្មវិធី។ ភាសាទាំងពីរមានលក្ខណៈពិសេស និងការប្រើប្រាស់ផ្សេងៗគ្នា ប៉ុន្តែត្រូវបានប្រើសម្រាប់ការអភិវឌ្ឍកម្មវិធីទូទៅ និងកម្មវិធីដែលមានលក្ខណៈពិសេស ។ ដោយសារតែសមត្ថភាព និងការគាំទ្រពីសហគមន៍ អ្នកអភិវឌ្ឍន៍ជា

ច្រើនបានសួរថា ភាសាណាមួយនឹងល្អជាងសម្រាប់គម្រោងណាមួយ។ ដូច្នេះការសិក្សាប្រៀបធៀបនេះត្រូវបានធ្វើឡើងដើម្បីបង្ហាញពីភាពខុសគ្នា និងសមាមាត្រនៃសមត្ថភាពរបស់ C# និង Java។

តើភាសាទាំងពីរភាសាមួយណាដែលមានសមត្ថភាព និងភាពងាយស្រួលក្នុងការសរសេរកូដជាង ?

៣. គោលបំណងនៃការស្រាវជ្រាវ

ក្នុងការចង់លើកស្ទួយនៅវិស័យបច្ចេកវិទ្យា និងផ្តល់នៅការបកស្រាយ ពន្យល់ណែនាំពីភាសាសរសេរកម្មវិធីទាំងពីរគឺ C# និង Java។ ហើយបង្ហាញពីការវិវត្តន៍របស់ភាសាសរសេរកម្មវិធីទាំងពីរនេះ។ និងដើម្បីសិក្សាភាពខុសគ្នា និងសមរម្យនៃការប្រើប្រាស់ C# និង Java ក្នុងការអភិវឌ្ឍកម្មវិធី។ ហើយនិងបង្ហាញពីលក្ខណៈពិសេស សមត្ថភាព និងភាពងាយស្រួលនៃភាសាទាំងពីរ រួចផ្តល់នូវមូលដ្ឋានសម្រាប់អ្នកអភិវឌ្ឍន៍ក្នុងការជ្រើសរើសភាសាណាមួយដែលសមរម្យសម្រាប់គម្រោងជាក់លាក់របស់ពួកគេ។ តាមរយៈការណែនាំនៃភាសាសរសេរកម្មវិធី និងការវិភាគស៊ីជម្រៅនៃភាសាសរសេរកម្មវិធីចំនួនពីរដែលបានជ្រើសរើសគឺ C# និង Java ពីភាពជាក់ស្តែង និក្ខេបបទនេះបង្ហាញពីគំនិតទូទៅអំពីភាសាសរសេរកម្មវិធីទាំងមូល ការយល់ដឹងស៊ីជម្រៅនៅក្នុងលក្ខណៈបច្ចេកទេស និងការអនុវត្តជាក់ស្តែង ក្នុងចំណោមភាសាសរសេរកម្មវិធីដែលបានជ្រើសរើសទាំងពីរនេះ។ គោលបំណងសំខាន់នៃនិក្ខេបបទនេះគឺដើម្បីណែនាំមនុស្សឱ្យធ្វើការសម្រេចចិត្តជាយុទ្ធសាស្ត្រក្នុងការប្រើប្រាស់ភាសាកម្មវិធីដែលសមស្របបំផុតដើម្បីបង្កើតប្រព័ន្ធកម្មវិធីថ្មី។

៤. ទំហំ និង ដែនកំណត់នៃការស្រាវជ្រាវ

ការធ្វើអ្វីមួយវាតែងតែមាននៅកម្រិតព្រំដែនរបស់វា ហើយសមត្ថភាពរបស់ក្រុមយើងខ្ញុំក៏នៅមានកំរិតនៅឡើយ រួមទាំងពេលវេលានៅមានកំរិតផងដែរនោះ ទើបក្រុមយើងខ្ញុំនិងធ្វើការបកស្រាយ និងវិភាគទៅលើបញ្ហាមួយចំនួនដូចជា៖

- ណែនាំបង្ហាញពីភាសាសរសេរកម្មវិធីពីរគឺ Java និងC#
- បង្ហាញពីការវិវត្តន៍ពីមួយជំនាន់ទៅមួយជំនាន់នៃភាសាទាំងពីរ
- ប្រៀបធៀបទៅលើវេយ្យាករណ៍មួយចំនួនរបស់ភាសាទាំងពីរ និងលក្ខណៈពិសេសរបស់វា
- ប្រៀធៀបទៅលើដំណើរការនៃការអនុវត្តន៍ជាក់ស្តែង នៃភាសាសរសេរកម្មវិធីទាំងពីនេះ
- ភាសាសរសេរកម្មវិធីផ្សេងទៀតមិនត្រូវបានពិភាក្សា និងប្រៀបធៀបនៅក្នុងនិក្ខេបបទនេះទេ។

៥. សារៈសំខាន់នៃការស្រាវជ្រាវ

ក្រោយពីបានសិក្សាយ៉ាងស៊ីជម្រៅទៅលើប្រធានបទមួយនេះវាពិតជាមានសារៈសំខាន់ណាស់សម្រាប់អ្នកដែលចង់សិក្សាទៅលើភាសារសរសេរកម្មវិធីដែលធ្វើអោយពួកគេទទួលបាននៅអត្ថប្រយោជន៍ដូចជា

- ធ្វើអោយពួកគេស្គាល់នៅភាសាសរសេរកម្មវិធី Java និង C# កាន់តែច្បាស់
- ស្គាល់ពីភាពខុសគ្នា និងដូចគ្នានៃភាសាទាំងពីរ ជាមួយនិងការវិវត្តន៍របស់ជំនាន់នីមួយៗនៃភាសាទាំងពីរ
- ជួយអោយសិស្ស និងនិស្សិតមានភាពងាយស្រួលសម្រេចចិត្ត ក្នុងការជ្រើសរើសភាសាមួយមកសិក្សា ឬបំពេញការងារតាមតម្រូវការជាក់ស្តែង។

៦. វិធីសាស្ត្រនៃការស្រាវជ្រាវ

នៅមុនពេលដែលពួកយើងបានធ្វើការពិភាក្សាគ្នាទៅលើប្រធានបទមួយនេះ ពួកយើងបានគិតទៅដល់នៅបញ្ហាធំមួយដែលបញ្ហានោះគឺជាការប្រមូលនៅទិន្នន័យផ្សេងៗដែលពាក់ព័ន្ធទៅនិងប្រធានបទ។ ដោយឡែកទិន្នន័យត្រូវបានបែងចែកជាពីរ គឺ primary data និង secondary data។ដែល

- ទិន្នន័យចម្បង (primary data) ជាទិន្នន័យដែលបានមកពីរ៖
    - ការសាកសួរទៅកាន់លោកគ្រូអ្នកគ្រូនៅក្នុងសាលា
    - ការសាកសួរទៅកាន់រៀមច្បង
- ទិន្នន័យបន្ទាប់បន្សំ(secondary data) ជាទិន្នន័យដែលបានមកពី៖
    - ការស្វែងរកនៅលើអ៊ីនធឺណែត
    - ឯកសាររបស់លោកគ្រូអ្នកគ្រូ ឬសិស្សច្បងដែលបានបន្សល់ទុកនៅបណ្ណាល័យ។

ជំពូកទី១

រំលឹកទ្រឹស្តី

១.១ ប្រវត្តិសង្ខេបនៃភាសាសរសេរកម្មវិធី

១.១.១ អ្វីទៅជាភាសាសរសេរកម្មវិធី?

ភាសាសរសេរកម្មវិធីគឺជាភាសាសិប្បនិម្មិតដែលត្រូវបានរចនាឡើងដើម្បីប្រើសម្រាប់សរសេរកម្មវិធី បញ្ជាកុំព្យូទ័រ ឬឧបករណ៍ម៉ាស៊ីនបច្ចេកវិទ្យាផ្សេងៗ ដើម្បីអនុវត្តមុខងារផ្សេងៗ។ ភាសាសរសេរកម្មវិធីបានអភិវឌ្ឍយ៉ាងលឿនតាំងពីដើមឆ្នាំ 1950 ដែលនាំទៅដល់ជាងរាប់រយនៃភាសាសរសេរកម្មវិធីផ្សេងៗគ្នាកំពុងត្រូវបានបង្កើត។ ជាមួយនឹងល្បឿនលឿនការអភិវឌ្ឍន៍បច្ចេកវិទ្យានៅក្នុង hardware ជាមួយនឹងprocessors លឿនជាងមុនកាន់តែច្រើនភាសាសរសេរកម្មវិធីដ៏មានឥទ្ធិពលអាចត្រូវបានប្រើដើម្បីបំពេញតម្រូវការនៃការរចនាបន្ថែមទៀតកម្មវិធីដែលមានប្រសិទ្ធភាពសម្រាប់កម្មវិធីផ្សេងៗ។ នៅក្នុងជំពូកនេះប្រាំជំនាន់នៃភាសាសរសេរកម្មវិធីត្រូវបានពិពណ៌នាដោយសង្ខេបតាមលំដាប់ពេលវេលា។

១.១.២ អ្វីទៅជាភាសាម៉ាស៊ីន ?(Machine languages)

ជំនាន់ដំបូងនៃភាសាសរសេរកម្មវិធីគឺជាភាសាម៉ាស៊ីនដែលលេចឡើងក្នុងដើមទសវត្សរ៍ឆ្នាំ 1950 ។ វាត្រូវបានសរសេរជាគោលពីរ ដែលជាស៊េរីនៃលេខសូន្យ និងលេខមួយ។ Binary គឺពិបាកណាស់ក្នុងការយល់ពីមនុស្ស និងងាយមានកំហុស។ សម្រាប់បញ្ហាចម្បងរបស់ភាសាម៉ាស៊ីនគឺអាស្រ័យលើម៉ាស៊ីន ពីព្រោះ “ភាសាម៉ាស៊ីនត្រូវបានបង្កើតឡើងខុសគ្នាសម្រាប់ស៊ីភីយូនីមួយៗ”។ ភាសាសរសេរកម្មវិធីដែលយើងប្រើសម្រាប់សរសេរកូដដូចជា C, C++ និង java គឺជាភាសាកម្រិតខ្ពស់។ ភាសាកម្រិតខ្ពស់មិនត្រូវបានយល់ដោយកុំព្យូទ័រដោយផ្ទាល់ទេ ដូច្នេះវាត្រូវបានបំប្លែងទៅជាភាសាម៉ាស៊ីនកម្រិតទាប ដើម្បីយល់ពីអត្ថន័យនៃកូដ និងអនុវត្តការប្រតិបត្តិ។ កុំព្យូទ័រចងក្រងកូដដែលសរសេរដោយយើង ហើយបកប្រែជាកូដម៉ាស៊ីន រួចប្រតិបត្តិវា។ កុំព្យូទ័រអាចយល់បានតែភាសាម៉ាស៊ីនប៉ុណ្ណោះ។

១.១.៣ អ្វីទៅជាភាសាសន្និបាតជានិមិត្ត?(Symbolic assembly languages)

ជាភាសាជំនាន់ទី 2 នៃភាសាសរសេរកម្មវិធីជាតំណាងនិមិត្តសញ្ញានៃសេចក្តីណែនាំម៉ាស៊ីន ការចុះឈ្មោះ និងអាសយដ្ឋានអង្គចងចាំអនុញ្ញាតឱ្យអ្នកសរសេរកម្មវិធីបង្កើតកម្មវិធីដែលអាចអានបានដោយមនុស្ស។ ដើម្បីអោយកុំព្យូទ័រយល់ពីកម្មវិធី វាត្រូវតែបំលែងទៅជាទម្រង់ម៉ាស៊ីនដែលអាចអានបានដោយប្រើ Assembler។ នៅក្នុងការសរសេរកម្មវិធីកុំព្យូទ័រ ភាសា assembly (ជាភាសា assembler ជាជម្រើស ឬកូដម៉ាស៊ីននិមិត្តសញ្ញា)

ដែលជារឿយៗសំដៅទៅសាមញ្ញថា assembly និងជាអក្សរកាត់ជាទូទៅថា ASM ឬ asm គឺជាភាសាសរសេរកម្មវិធីកម្រិតទាបណាមួយដែលមានការឆ្លើយឆ្លងគ្នាខ្លាំងរវាងការណែនាំជាភាសា និង ស្ថាបត្យកម្ម។ហើយភាសានេះសរសេរក្នុងទម្រង់សាមញ្ញជាង និងនៅកម្រិតខ្ពស់នៃភាសាម៉ាស៊ីន abstraction ជំនួសឱ្យស៊េរីលេខសូន្យ ហើយមួយមាននិមិត្តសញ្ញា (ភាគរយ ប្រាក់ដុល្លារ និងផ្នែកនៃបន្សំពាក្យ និងលេខដែលបានប្រើធ្វើការបញ្ជា។ ប៉ុន្តែភាសាសរសេរកម្មវិធីការជួបប្រជុំជានិមិត្តរូបនៅតែមានបញ្ហានៃភាពអាស្រ័យផ្នែករឹងខ្ពស់ និងកង្វះនៃការចល័ត ដែលមានន័យថាកូដជួបប្រជុំគ្នាមិនអាចអនុវត្តនៅលើម៉ាស៊ីនផ្សេងគ្នាបានទេ។

១.១.៤ អ្វីទៅជាភាសាដែលមានបញ្ហា ? (Problem-oriented languages)

រយៈពេលពីដើមទសវត្សរ៍ឆ្នាំ 1960 ដល់ឆ្នាំ 1980 បាននាំយើងមកនូវកម្មវិធីភាសាជំនាន់ទីបី។ ជំនាន់ទីបីនៃភាសាសរសេរកម្មវិធីត្រូវបានគេហៅថាបញ្ហាតម្រង់ទិសភាសា និងត្រូវបានចាត់ទុកថាជាភាសាកម្រិតខ្ពស់ផងដែរ។ ជំនាន់ទីបីភាសាសរសេរកម្មវិធីត្រូវបានបំប្លែងពីភាសាអង់គ្លេសទៅជាភាសាម៉ាស៊ីន កម្មវិធីចងក្រងត្រូវបានប្រើដើម្បីបំប្លែងការណែនាំទាំងនេះ។ C, C++, រយៈពេលពីដើមទសវត្សរ៍ឆ្នាំ 1960 ដល់ឆ្នាំ 1980 បាននាំយើងមកនូវកម្មវិធីភាសាជំនាន់ទីបី។ ជំនាន់ទីបីនៃភាសាសរសេរកម្មវិធីត្រូវបានគេហៅថាបញ្ហាតម្រង់ទិសភាសា និងត្រូវបានចាត់ទុកថាជាភាសាកម្រិតខ្ពស់ផងដែរ។ ជំនាន់ទីបីភាសាសរសេរកម្មវិធីត្រូវបានបំប្លែងពីភាសាអង់គ្លេសទៅជាភាសាម៉ាស៊ីន កម្មវិធីចងក្រងត្រូវបានប្រើដើម្បីបំប្លែងការណែនាំទាំងនេះ។

១.១.៥ អ្វីទៅជាភាសាមិនមែននីតិវិធី ? (Non-procedural languages)

ភាសាសរសេរកម្មវិធីជំនាន់ទី 4 គឺផ្តោតលើការដោះស្រាយបញ្ហា។ ភាពខុសគ្នាសំខាន់រវាងភាសាសរសេរកម្មវិធីជំនាន់ផ្សេងទៀតគឺថាពួកគេខ្វល់ខ្វាយនឹងអ្វីដែលត្រូវធ្វើ ជាងរបៀបពិតប្រកដ។ "លក្ខណៈពិសេសដែលបង្ហាញអោយឃើញនៅក្នុងភាសាជំនាន់ទី 4 ច្បាស់ណាស់គឺថាវាត្រូវតែងាយស្រួលប្រើ ចល័ត និងឯករាជ្យនៃប្រព័ន្ធប្រតិបត្តិការ ដែលអាចប្រើបានដោយអ្នកមិនសរសេរកម្មវិធី មានជម្រើសលំនាំដើមឆ្លាតវៃអំពីអ្វីដែលអ្នកប្រើចង់បាន និងអនុញ្ញាតឱ្យអ្នកប្រើទទួលបានលទ្ធផលលឿនដោយប្រើតម្រូវការអប្បបរមា។កូដដែលបង្កើតដោយលេខកូដគ្មានកំហុសពីកន្សោមកម្រិតខ្ពស់(ប្រើមូលដ្ឋានទិន្នន័យ និងការគ្រប់គ្រងវចនានុក្រមដែលធ្វើអោយកម្មវិធីងាយស្រួល និងរហ័សក្នុងការផ្លាស់ប្តូរ)។ ឧទាហរណ៍ដ៏ល្បីបំផុតនៃភាសាសរសេរកម្មវិធីជំនាន់ទីបួនគឺ SQL, MYSQL ។

នៅទសវត្សរ៍ឆ្នាំ 1980 និងឆ្នាំ 1990 មានកិច្ចខិតខំប្រឹងប្រែងដើម្បីបង្កើតភាសាកម្មវិធីជំនាន់ទីប្រាំ (5GL) ។

១.១.៦ អ្វីទៅជាភាសាសរសេរកម្មវិធីជំនាន់ទីប្រាំ ? (Fifth-generation programming languages)

ភាសាសរសេរកម្មវិធីជំនាន់ទីប្រាំ (អក្សរកាត់ 5GL) គឺជាការសរសេរកម្មវិធីផ្អែកលើការដោះស្រាយបញ្ហា ដោយប្រើឧបសគ្គដែលបានផ្តល់ឱ្យកម្មវិធីជាជាងប្រើក្បួនដោះស្រាយដែលសរសេរដោយអ្នកសរសេរកម្មវិធី។ ជាមួយនឹងការប្រើប្រាស់ទីប្រាំជំនាន់នៃភាសាសរសេរកម្មវិធី កុំព្យូទ័រអាចមានសមត្ថភាពគិតរៀងៗខ្លួនហើយការសន្និដ្ឋានផ្ទាល់ខ្លួនរបស់ពួកគេអាចត្រូវបានដោះស្រាយដោយប្រើព័ត៌មានដែលបានរៀបចំកម្មវិធីក្នុងមូលដ្ឋានទិន្នន័យធំ។ ក្តីសុបិន្តរបស់មនុស្សយន្តដែលមានបញ្ញាសិប្បនិម្មិត និងតក្កវិជ្ជាមិនច្បាស់បានមកដល់ពិតដោយប្រើភាសាសរសេរកម្មវិធីជំនាន់នេះ។

១.២ ប្រវត្តិរបស់ភាសា Java និង C#

១.២.១ អ្វីទៅជាភាសាJava ?

Java គឺជាភាសមួយដែលត្រូវបានបង្កើតឡើងដោយលោក James Gosling ។ គាត់ជាបុគ្គលិកធ្វើការនៅក្រុមហ៊ុន Sun Microsystems នៅក្នុងគំរោងស្រាវជ្រាវមួយឈ្មោះថា Green Project នៅឆ្នាំ ១៩៩១។ គម្រោងនេះត្រូវបានប្រើភាសាមួយដែលមានមូលដ្ឋានឈរលើភាសា C និង C++ ។ ដំបូងឡើយភាសានេះឈ្មោះថា oak បន្ទាប់ពីលោកបានឃើញដើម oak តាមរយៈ បង្អួចការិយាល័យរបស់គាត់ក្នុងក្រុមហ៊ុម Sun។ តែឈ្មោះនេះត្រូវបានប្តូរតាមសំណើររបស់មិត្តរួមការងាររបស់គាត់ទៅជា Java បន្ទាប់ពីគាត់ត្រឡប់ពីហាងកាហ្វេ ហើយក៏មានឈ្មោះថា Java មកដល់សព្វថ្ងៃនេះ។

ការងាររបស់ Green Project មានការលំបាកយ៉ាងខ្លាំងជាហេតុធ្វើអោយកិច្ចព្រមព្រៀងមួយរបស់ក្រុមហ៊ុន Sun ត្រូវបានប្រគល់អោយក្រុមហ៊ុនដ៏ទៃ។ ការងាររបស់ Green Project ស្ទើតែដួលរលំទៅហើយ តែសំណាងល្អនៅឆ្នាំ ១៩៩៣ World Wide Web បានលេចធ្លោឡើងបានធ្វើអោយអ្នកធ្វើការនៅក្រុមហ៊ុន Sun មើលឃើញយ៉ាងច្បាស់ពីអនុភាពនៃការប្រើ Java ដើម្បីបង្កើត Web Page វាបានធ្វើអោយគម្រោងនេះដើរសារជាថ្មីវិញ។

- Version របស់ Java

- Java 1.0 (1996):កំណែដើមនៃ Java ដែលបានដាក់បង្ហាញជាផ្លូវការទៅកាន់សាធារណៈដែលជាការចេញផ្សាយដំបូងបង្អស់ ដែលសំដៅលើគោលការណ៍ "សរសេរម្តង ប្រើបានគ្រប់ទីកន្លែង" (Write Once, Run Anywhere)។

- Java 1.1 (1997):បន្ថែមលក្ខណៈពិសេសដូចជា JDBC (Java Database Connectivity), inner classes, RMI (Remote Method Invocation) ព្រមទាំងមានការកែលម្អសមត្ថភាព API និង AWT (Abstract Window Toolkit)។
- Java 2 (1.2 - 1.4) (1998 - 2002):Java 2 ដាក់បង្ហាញជាកំណែធំបំផុតមួយដែលបង្កើតពិភពថ្មីសម្រាប់ Java និង ធ្វើឱ្យប្រសើរឡើងដោយបន្ថែម Swing សម្រាប់ GUI និង Collections Framework។
- Java 5 (1.5) (2004):កំណែនេះមានការកែលម្អពាក់ព័ន្ធនឹងការដំណើរការខ្ពស់និងឧបករណ៍បង្កើតកម្មវិធី (development tools) ដោយបន្ថែម Generics, Enums, Annotations, និង Enhanced for-loops។
- Java 6 (2006):ធ្វើឱ្យប្រសើរនូវប្រសិទ្ធិភាព និងគាំទ្រការសរសេរស្គ្រីបតាមរយៈ JVM, បញ្ចូលនូវ XML, Log API, និងសុវត្ថិភាពសម្រាប់ Web Applications។
- Java 7 (2011):ជាបច្ចុប្បន្នភាពធំដែលបន្ថែម Generics, Metadata, Enum, បញ្ចូល Project Coin, NIO2 (ប្រព័ន្ធ I/O), និង Try-With-Resources, និងសកម្មភាពផ្សេងៗ។
- Java 1.8 (2014):កំណែនេះផ្តោតលើសមត្ថភាពខ្ពស់និងបរិស្ថានអភិវឌ្ឍ Eclipse IDE និង ចេញផ្សាយ Lambda Expressions, Stream API, និង Date/Time API ថ្មី។
- Java 1.9 (2017):ជាកំណែធំបន្ទាប់ បន្ថែម Try-With-Resources Statement និង Fork/Join Framework, បញ្ចូល Modular System (Project Jigsaw) និង JShell (REPL)។
- Java 10 (2018):កំណែដែលមាន Lambda Expressions និង Stream API ដែលគេប្រើសម្រាប់ការគ្រប់គ្រងដាក់គ្នាច្រើននៃទិន្នន័យ បន្ថែម var សម្រាប់ការបង្កើតអថេរផ្លាស់ប្តូរទំហំប្រភេទ។
- Java 11 (2018):ជា Long-Term Support (LTS) ចុងក្រោយមួយ បូករួមនឹងការដក JavaFX ចេញពី JDK មូលដ្ឋាន ដាក់បញ្ចូល Modularity និង JShell (REPL)។
- Java 12-15 (2019-2020):កំណែខ្លីបង្អស់នេះមានការផ្លាស់ប្តូរក្នុងការគ្រប់គ្រងពុំភ្លូសន់ និង Type Inference (var) បង្កើតអនុសាសន៍ដូចជា Switch Expressions និង Text Blocks។

- Java 17 (2021):កំណែដែលបន្ថែមនូវការគាំទ្រលើ Unicode 10 និងសមត្ថភាពក្នុងការដាក់ដំណើរការ Java file ផ្ទាល់ជា LTS ថ្មីបន្ទាប់ពី Java 11 ដែលរួមបញ្ចូល Sealed Classes និងការធ្វើប្រសើរដទៃទៀត។
- Java 18 (2022):ជាកំណែ LTS (Long-Term Support) បច្ចុប្បន្នដែលបន្ថែមការកែលម្អស្ថិរភាពនិងសមត្ថភាព ការធ្វើបច្ចុប្បន្នភាពតូចៗ ដែលមានការកំណត់ UTF-8 ជាភាសាមានលំនាំដើម។
- Java 19 (2022):កំណែចុងក្រោយបំផុតដែលបន្ថែមនូវសមត្ថភាពថ្មីៗសម្រាប់ដំណើរការរលឿននិងការបង្កើតកម្មវិធីលើ Web លក្ខណៈពិសេសសាកល្បងដូច Threads ដែលអាចធ្វើបាន Virtual Threads និង Structured Concurrency។
- Java 20 (2023):ធ្វើឱ្យប្រសើរការក្សាទុកប្រសិទ្ធិភាព និងធ្វើស្ថេរភាពលក្ខណៈពិសេសសាកល្បង។

➢ លក្ខណៈពិសេសរបស់ Java

គេថា Java មានលក្ខណៈ Simple, Object Oriented, Statically Typed, Compiled and Interpreted, Architecture Neutral and Portable, Multithreaded, Garbage Collected, robust, Secure, Built-in Networking and Extensible ។

- Simple : អ្នកបង្កើត Java បានលុបបំបាត់ចោលនូវលក្ខណៈមិនចាំបាច់ចំនួនរបស់ភាសាសរសេរកម្មវិធី ដូចជា Java មិនប្រើ Pointers, Structures, Unions, Templates, Header file ឬ Multiple Inheritance ជាដើម ។
- Object Oriented : ដូចនឹងភាសា C ឬ C++ ដែរ គឺប្រើ Classes ដើម្បីរៀបចំ Code ឲទៅជាសំនុំមួយត្រូវហើយវាបង្កើត Objects តាមរយៈ Classes ។ Class របស់ Java អាចទទួលលក្ខណៈពី Class មួយផ្សេងទៀត ។ ប៉ុន្តែ Class មួយមិនអាចទទួលលក្ខណៈពី Classes ច្រើនបានឡើយ ។
- Statically Typed : គ្រប់ Object ទាំងអស់មុនពេលប្រើប្រាស់នៅក្នុងកម្មវិធីមួយ ដាច់ខាតត្រូវប្រកាសវាជាមុនសិន។ លក្ខណៈនេះអាចធ្វើឲ Compiler របស់ Java រកឃើញនៅទីតាំងនិងប្រាប់ឲដឹងនៅប្រភេទទិន្នន័យដែលមិនត្រូវគ្នា ។
- Compiled and Interpreted : មុននឹងយើងអាចដំណើរការកម្មវិធីមួយដែលសរសេរឡើងដោយភាសា Java បាននោះលុះត្រាតែយើង Compile វាតាមរយៈ Compiler របស់វាជាមុនសិន។ នៅពេល

Compile រួចដោយជោគជ័យ វានឹងបង្កើត File មួយផ្សេងទៀតប្រភេទជា Byte-Code ដែលស្រដៀងគ្នាទៅនឹង Machine-Code ហើយវាអាចដើរការក្នុងប្រព័ន្ធ Computer ដោយមាន Interpreter របស់ Java ។ Interpreter ជាអ្នកបកប្រែពី Byte-code ទៅជាពាក្យបញ្ជាប្រភេទ machine-code ។ ហេតុនេះហើយបានជាគេថា Java មានលក្ខណៈ Compiled and Interpreted ។

- Architecture Neutral and Portable : ដោយសារកម្មវិធីដែលសរសេរឡើងដោយ Java ត្រូវបាន Compile ជាទម្រង់ Byte-code ដែលមានលក្ខណៈមិនអាស្រ័យនឹងទម្រង់ខាងក្នុងរបស់ Computer ជាហេតុធ្វើឲកម្មវិធីដែលសរសេរឡើងដោយ Java អាចដំណើរការលើប្រព័ន្ធណាមួយក៏បាន(មិនប្រកាប់ plate form) ។
- Multithreaded : Java មាន threads សម្រាប់អនុវត្តធ្វើប្រតិបត្តិការងារច្រើនក្នុងពេលតែមួយបាន។
- Garbage Collected : Java បានធ្វើការប្រមូលនូវអ្វីៗដែលមិនចាំបាច់ លុបចោលពី Memory ដោយខ្លួនឯងមិនចាំបាច់សរសេរ Code ដើម្បីលុប Object ដែលមិនប្រើនោះទេ(បានន័យថា Variable ឬ Objects ណាដែលឈប់ប្រើហើយវានឹងលុបចោលដោយខ្លួនឯង ជួយសម្រួលដល់អ្នកសរសេរកម្មវិធីមិនព្រួយបារម្ភណ៍ពីការខ្វះខាត Memory ដោយសារ Objects មិនបានការណ៍នោះឡើយ) ។
- Robust : ដោយសារ Interpreter របស់ Java ពិនិត្យគ្រប់ដំណើរការចូលទៅក្នុងប្រព័ន្ធទាំងអស់របស់កម្មវិធី ជាហេតុធ្វើឲកម្មវិធីសរសេរដោយ Java មិនប៉ះពាល់ដល់ប្រព័ន្ធ Computer ឡើយ។ កាលណាវាមាន Error វានឹងបង្កើតជា Exception ។
- Secure : វាមិនគ្រាន់តែត្រួតពិនិត្យរាល់ដំណើរការការចូលក្នុង Memory ប៉ុណ្ណោះទេ វាថែមទាំងធានាមិនឆ្លង Virus នៅពេលកំពុងដំណើរការកម្មវិធីទៀតផង ព្រោះវាមិនប្រើ Pointer ធ្វើឲ Virus មិនអាចចូលទៅកាន់ Memory នៃប្រព័ន្ធ Computer បានឡើយ ។
- Built-in Networking : Java បានបង្កើតឲមានការប្រើលក្ខណៈជាបណ្តាញ ដោយនាំមកនូវ Classes ជាច្រើនសម្រាប់បង្កើតទំនាក់ទំនងជាមួយ Internet ។
- Extensible : Java អាចឲប្រើនូវ native methods ដែលជា Functions ដែលសរសេរឡើងក្នុងភាសាផ្សេងៗ ដូចជា C ឬ C++ ជាដើម។ លក្ខណៈនេះវាធ្វើឲអ្នកសរសេរកម្មវិធីសរសេរ function ដែលអាចប្រតិបត្តិការបានលឿនជាងការសរសេរ functions នៅក្នុង Java។ Native Methods ដំណើរការភ្ជាប់ទៅនឹងកម្មវិធី Java មានន័យថាវាបញ្ចូលជាមួយកម្មវិធីនៅក្នុងពេលដំណើរការកម្មវិធី។ នៅ

ពេលដែល Java ត្រូវបានជម្រុញលើបញ្ហាល្បឿននោះ Native Methods ប្រហែលជាឥតត្រូវការទៀតឡើយ។ លើសពីនេះ Java អាចឲគេប្រើជាមួយនឹងកម្មវិធីផ្សេងទៀតបាន ដូចជា Microsoft Access និង HTML ជាដើម ។

១.២.២ អ្វីទៅជាភាសាC#?

C# គឺជាភាសារសេរកម្មវិធីកម្រិតខ្ពស់ ទំនើប និងតម្រង់ទិស ដែលត្រូវបានបង្កើតឡើងដោយ Microsoft ។ វាត្រូវបានបង្កើតឡើងដើម្បីផ្តល់នូវភាសាសាមញ្ញ ទំនើប ដ៏មានអានុភាព សុវត្ថិភាព និងភាសាតម្រង់ទិសវត្ថុសម្រាប់ការអភិវឌ្ឍកម្មវិធី និងកម្មវិធី Windows សម្រាប់ក្របខ័ណ្ឌ .NET ដែលអាចត្រូវបានប្រើសម្រាប់បង្កើតកម្មវិធីជាច្រើន ចាប់ពីកម្មវិធីកុំព្យូទ័រលើតុ រហូតដល់សេវាកម្មគេហទំព័រ ។ C# ត្រូវបានបង្កើតឡើងនៅក្នុងខែមករា ឆ្នាំ 1999 ដែលអ្នកបង្កើតដ៏សំខាន់នៃភាសារសេរកម្មវិធីនេះគឺ Anders Hejlsberg, Scott Wiltamuth, និង Peter Golde មកពី Microsoft។ Anders Hejlsberg បានបង្កើតក្រុមដើម្បីបង្កើតភាសាថ្មីមួយនៅពេលនោះហៅថា Cool ដែលតំណាងឱ្យ "ភាសា C-like Object Oriented Language"។ ក្រុមហ៊ុន Microsoft បានពិចារណារក្សាឈ្មោះ "Cool" ជាឈ្មោះចុងក្រោយនៃភាសា ប៉ុន្តែបានជ្រើសរើសមិនធ្វើដូច្នេះសម្រាប់ហេតុផលយីហោ។ នៅពេលដែលគម្រោង .NET ត្រូវបានប្រកាសជាសាធារណៈនៅឯសន្និសីទអ្នកអភិវឌ្ឍន៍វិជ្ជាជីវៈ (PDC) ខែកក្កដា ឆ្នាំ 2000 ភាសាត្រូវបានប្តូរឈ្មោះទៅជា C# ហើយបណ្ណាល័យថ្នាក់ និងម៉ោងដំណើរការ ASP.NET ត្រូវបានបញ្ជូនទៅ C# ។ វាត្រូវបានចេញផ្សាយជាផ្នែកមួយនៃកំណែដំបូងនៃ .NET Framework រួមជាមួយ Visual Studio .NET ក្នុងឆ្នាំ 2002 ។ ភាសានេះត្រូវបានរចនាឡើងដើម្បីឱ្យមានសុវត្ថិភាពប្រភេទ និងគ្រប់គ្រងអង្គចងចាំដោយស្វ័យប្រវត្តិ ជាមួយនឹងវាក្យសម្ព័ន្ធដែលធ្លាប់ស្គាល់សម្រាប់អ្នកអភិវឌ្ឍន៍ដែលប្រើ C++ ឬ Java ។

Anders Hejlsberg គឺជាអ្នករចនាដ៏សំខាន់របស់ C# និងជាស្ថាបត្យករនាំមុខគេនៅក្រុមហ៊ុន Microsoft ហើយពីមុនត្រូវបានចូលរួមជាមួយនឹងការរចនារបស់ Turbo Pascal, Embarcadero Delphi (អតីត CodeGear Delphi, Inprise Delphi និង Borland Delphi) និង Visual J++ ។ នៅក្នុងបទសម្ភាសន៍ និងឯកសារបច្ចេកទេស គាត់បាននិយាយថា កំហុស នៅក្នុងភាសារសេរកម្មវិធីសំខាន់ៗភាគច្រើន (ឧទាហរណ៍ C++, Java, Delphi និង Smalltalk) បានជំរុញមូលដ្ឋានគ្រឹះនៃ Common Language Runtime (CLR) ដែលជំរុញឱ្យការរចនា នៃភាសា C# ។

C# ធ្លាប់មាន mascot ឈ្មោះ Andy (ដាក់ឈ្មោះតាម Anders Hejlsberg)។ វាត្រូវបានចូលនិវត្តន៍នៅថ្ងៃទី 29 ខែមករា ឆ្នាំ 2004។

C# ដើមឡើយត្រូវបានដាក់ជូនអនុគណៈកម្មាធិការ ISO/IEC JTC 1 SC 22 ដើម្បីពិនិត្យ ក្រោម ISO/IEC 23270:2003 ត្រូវបានដកចេញ ហើយក្រោយមកត្រូវបានអនុម័តក្រោម ISO/IEC 23270:2006។ 23270:2006 ត្រូវបានដកចេញក្រោម 23270:2018 ហើយត្រូវបានអនុម័តជាមួយនឹងកំណែនេះ។

- ការវិវត្តន៍របស់ C# (Version)

C# (C-Sharp) គឺជាភាសាសរសេរកម្មវិធីដែលបង្កើតឡើងដោយ Microsoft ហើយវាបានឆ្លងកាត់កំណែផ្សេងៗចាប់តាំងពីការចេញផ្សាយដំបូងរបស់វា។ កំណែនីមួយៗណែនាំមុខងារថ្មីៗ និងការកែលម្អ។ នេះជាទិដ្ឋភាពទូទៅនៃកំណែ C# សំខាន់ៗ៖

- C# 1.0 (2002)៖ ត្រូវបានចេញផ្សាយរួមជាមួយ .NET Framework 1.0 ក្នុងឆ្នាំ 2002។ វាផ្តល់នូវលក្ខណៈពិសេសជាមូលដ្ឋានធម្មតានៃភាសាដែលតម្រង់ទិសវត្ថុ ដូចជាថ្នាក់ មរតក និងចំណុចប្រទាក់។ កំណែនេះបានដាក់មូលដ្ឋានគ្រឹះសម្រាប់អ្វីដែលនឹងក្លាយជាភាសាសរសេរកម្មវិធីដែលត្រូវបានប្រើប្រាស់យ៉ាងទូលំទូលាយបំផុត។
- C# 2.0 (2005)៖ ចេញផ្សាយជាមួយ .NET Framework 2.0 កំណែនេះបានណែនាំលក្ខណៈពិសេសសំខាន់ៗដូចជា generics, partial classes, nullable types, and anonymous method, ដែលបានបង្កើនសមត្ថភាពនិងភាពបត់បែនរបស់ភាសាយ៉ាងខ្លាំង។
- C# 3.0 (2007)៖ ចេញផ្សាយជាមួយ .NET Framework 3.5, C# 3.0 បានណែនាំ LINQ (Language Integrated Query) ដែលជាមុខងារបដិវត្តន៍ដែលអនុញ្ញាតឱ្យមានសំណួរទិន្នន័យពីប្រភពផ្សេងៗដោយផ្ទាល់នៅក្នុង C# ។ កំណែនេះក៏បាននាំយកកន្សោម lambda វិធីសាស្ត្របន្ថែម និងប្រភេទអនាមិក ធ្វើឱ្យភាសាកាន់តែទំនើបថែមទៀត។
- C# 4.0 (2010)៖ ចេញផ្សាយជាមួយ .NET Framework 4.0 កំណែនេះបានបន្ថែមការវាយអក្សរថាមវន្ត ការកំណត់ឈ្មោះ និងជម្រើស និងភាពផ្ទុយគ្នា និងភាពផ្ទុយគ្នានៅក្នុងប្រភេទទូទៅ ដែលធ្វើឱ្យភាសាកាន់តែមានភាពបត់បែន និងងាយស្រួលប្រើក្នុងសេណារីយ៉ូកម្មវិធីផ្សេងៗ។
- C# 5.0 (2012)៖ ជាមួយនឹងការចេញផ្សាយ .NET Framework 4.5, C# 5.0 បានណែនាំអសមកាល និងរង់ចាំពាក្យគន្លឹះ ដែលសម្រួលការសរសេរកម្មវិធីអសមកាល អនុញ្ញាតឱ្យអ្នកអភិវឌ្ឍន៍សរសេរកូដកាន់តែស្អាត និងអាចគ្រប់គ្រងបានជាងមុន នៅពេលដោះស្រាយជាមួយប្រតិបត្តិការអសមកាល។

- C# 6.0 (2015)៖ កំណែនេះដែលបានចេញផ្សាយជាមួយ .NET Framework 4.6 និង Visual Studio 2015 ផ្តោតលើការបង្កើនផលិតភាពរបស់អ្នកអភិវឌ្ឍន៍។ វាបានណែនាំលក្ខណៈពិសេសដូចជា auto-property initializers, expression-bodied member, null-conditional operators និង string interpolation ដែលធ្វើឱ្យកូដកាន់តែសង្ខេប និងអាចអានបាន។
- C# 7.0 - 7.3 (2017-2018)៖ ចេញជាមួយ Visual Studio 2017 និង .NET Core 1.0/2.0 កំណែទាំងនេះបានណែនាំ tuples ការផ្គូផ្គងលំនាំ មុខងារមូលដ្ឋាន និងអថេរចេញ។ លក្ខណៈពិសេសទាំងនេះបានពង្រឹងការបញ្ចេញមតិរបស់ភាសា និងអនុញ្ញាតឱ្អកអភិវឌ្ឍន៍សរសេរកូដកាន់តែតូច និងវិចារណញាណ ref returns, out variables និងច្រើនទៀត។
- C# 8.0 (2019)៖ ចេញផ្សាយជាមួយ .NET Core 3.0 និង Visual Studio 2019, C# 8.0 ណែនាំប្រភេទឯកសារយោងដែលមិនអាចកាត់ថ្លៃបាន ស្ទ្រីម async ជួរ សន្ទស្សន៍ និងកន្សោមប្តូរ។ លក្ខណៈពិសេសទាំងនេះមានគោលបំណងធ្វើឱ្យប្រសើរឡើងនូវសុវត្ថិភាពកូដ ដំណើរការ និងលទ្ធភាពអាន។
- C# 9.0 (2020)៖ ចេញផ្សាយជាមួយ .NET 5.0, C# 9.0 បាននាំមកក្នុងកំណត់ត្រា (ប្រភេទឯកសារយោងថ្មីសម្រាប់ទិន្នន័យដែលមិនអាចផ្លាស់ប្តូរបាន) អ្នកកំណត់តែនៅក្នុងវា សេចក្តីថ្លែងការកម្រិតកំពូល និងការផ្គូផ្គងលំនាំដែលប្រសើរឡើង ដោយបន្តនិន្នាការនៃការកែលម្អការបញ្ចេញមតិកូដ និងភាពមិនអាចផ្លាស់ប្តូរបាន។
- 10. C# 10.0 (2021)៖ ចេញផ្សាយជាមួយ .NET 6.0 ណែនាំជាសកលដោយប្រើការណែនាំ កន្លែងដាក់ឈ្មោះឯកសារ រចនាសម្ព័ន្ធកំណត់ត្រា និងការកែលម្អ lambdas ក្នុងគោលបំណងកាត់បន្ថយកូដ boilerplate និងបង្កើនការអនុវត្ត។
- C# 11.0 (2022)៖ ចេញផ្សាយជាមួយ .NET 7.0 , C# 11.0 រួមបញ្ចូលលក្ខណៈពិសេសដូចជាអក្សរដើម ខ្សែអក្សរ លំនាំបញ្ជី និងគុណលក្ខណៈទូទៅ ដោយផ្តោតលើការធ្វើឱ្យភាសាកាន់តែមានការបញ្ចេញមតិ និងងាយស្រួលសម្រាប់អ្នកអភិវឌ្ឍន៍។
- C# 12.0 (រំពឹងទុកក្នុងឆ្នាំ 2024)៖ កំណត់ដើម្បីចេញផ្សាយជាមួយ .NET 8.0, C# 12.0 ត្រូវបានគេរំពឹងថានឹងណែនាំការធ្វើឱ្យប្រសើរឡើងដល់កំណត់ត្រា អ្នកបង្កើតបឋម និងការកែលម្អភាសាផ្សេងទៀត ដោយបន្តការវិវត្តន៍របស់វាជាភាសាដែលអាចប្រើប្រាស់បាន និងមានឥទ្ធិពល។

ការវិវត្តជាបន្តបន្ទាប់នៃភាសាដែលជំរុញដោយតម្រូវការនៃការអភិវឌ្ឍន៍កម្មវិធីទំនើបបានធានានូវភាពពាក់ព័ន្ធ និងប្រជាប្រិយភាពរបស់វានៅក្នុងសហគមន៍សរសេរកម្មវិធី។ C# មិនត្រឹមតែជាផ្នែកសំខាន់នៃយុទ្ធសា

ស្ត្រអភិវឌ្ឍន៍របស់ Microsoft ប៉ុណ្ណោះទេ ប៉ុន្តែវាក៏ជាភាសាដ៏មានឥទ្ធិពលនៅក្នុងពិភពអភិវឌ្ឍន៍កម្មវិធីដ៏ធំទូលាយផងដែរ។

- លក្ខណៈពិសេសរបស់C#

C# មានលក្ខណៈពិសេសជាច្រើនដែលធ្វើឱ្យវាល្អប្រើប្រាស់សម្រាប់ការអភិវឌ្ឍន៍កម្មវិធី។ ខាងក្រោមនេះជាលក្ខណៈពិសេសសំខាន់ៗរបស់ C#៖

- អភិវឌ្ឍដោយ Microsoft: C# ត្រូវបានបង្កើតឡើងដោយ Microsoft ហើយត្រូវបានគាំទ្រដោយ .NET Framework ឬ .NET Core។
- Object-Oriented Programming (OOP): C# គឺជាភាសាដែលផ្អែកលើមូលដ្ឋាន OOP ដូចជា Classes, Objects, Inheritance, Encapsulation, Polymorphism, និង Abstraction ។
- Automatic Memory Management (Garbage Collection): C# មានប្រព័ន្ធ Garbage Collection ដែលគ្រប់គ្រងនិងបញ្ចេញអង្គចងចាំដោយស្វ័យប្រវត្តិ។
- Type Safety: C# ផ្តល់សុវត្ថិភាពក្នុងការចាត់ចែងទិន្នន័យ (Type Safety) ដោយការពារការចុះចតនៃកូដឆ្គាំឆ្គង។
- Multithreading Support: C# មានលក្ខណៈសមត្ថភាពដើម្បីបង្កើត Thread ផ្សេងៗ សម្រាប់ការងារពាក់ព័ន្ធ Multithreading ដែលជួយពង្រីកប្រសិទ្ធភាពកម្មវិធី។
- LINQ (Language Integrated Query): LINQ ជាមធ្យោបាយងាយស្រួលក្នុងការសួរទិន្នន័យពី Collections, Databases, XML, ឬវត្ថុផ្សេងៗ។
- Asynchronous Programming: C# ផ្តល់នូវការគាំទ្រសម្រាប់ការប្រើប្រាស់ async និង await ដែលជួយធ្វើឱ្យការអភិវឌ្ឍកម្មវិធីអាចដំណើរការការងារតាមរយៈ Asynchronous I/O បានយ៉ាងមានប្រសិទ្ធភាព។
- Cross-platform Development: ជាមួយនឹង .NET Core, C# អាចប្រើប្រាស់សម្រាប់ការអភិវឌ្ឍកម្មវិធីលើប្រព័ន្ធដំណើរការដូចជា Windows, macOS, និង Linux។
- Interoperability: C# អាចអន្តរប្រតិបត្តិជាមួយភាសាផ្សេងៗដែលដំណើរការលើ .NET Framework ឬ COM objects។
- Rich Library Support: C# មានក្រុមហ៊ុន Libraries និង APIs ផ្ទាល់ពី .NET ដែលផ្តល់នូវមុខងារដើម្បីអភិវឌ្ឍកម្មវិធីប្រកបដោយគុណភាពខ្ពស់។

ជំពូកទី២

ប្រៀបធៀប

២.១ Data Type និង Size

Data type (ប្រភេទទិន្នន័យ) គឺជាប្រភេទនៃទិន្នន័យ ដែលប្រើប្រាស់ក្នុងកម្មវិធី Programming ដើម្បីកំណត់អំពីទិន្នន័យមួយចំនួន ហើយកំណត់ថា ត្រូវបានរក្សាទុកយ៉ាងដូចម្តេច និងអាចធ្វើសកម្មភាពអ្វីខ្លះចំពោះវា។ តម្លៃនីមួយៗនៅក្នុងកម្មវិធីមាន data type ផ្ទាល់ខ្លួន ដែលអាចជាប្រភេទទិន្នន័យជាមូលដ្ឋាន (primitive types) ឬ ប្រភេទទិន្នន័យដែលបានកំណត់ផ្ទាល់ (user-defined types)។

Size ជាទំហំអង្គចងចាំ (memory) ដែលត្រូវបានទាមទារដោយ Data type នីមួយៗ ដើម្បីរក្សាទុកតម្លៃមួយ។ ទំហំដែលប្រើប្រាស់ត្រូវបានវាស់វែងជាធម្មតាជាប៊ីត (bit) ឬបៃ (byte)។

- Data type របស់ Java

ក្នុងភាសា Java, data type (ប្រភេទទិន្នន័យ) ជាអ្វីដែលបញ្ជាក់អំពីទំហំ និងប្រភេទនៃតម្លៃដែលអាចបានទុកនៅក្នុងអថេរមួយ។ Java គឺជាភាសាដែលមានការកំណត់ប្រភេទមុន (statically typed language) ដែលត្រូវចាំបាច់ឱ្យគ្រប់អថេរត្រូវបានប្រកាសជាមួយប្រភេទទិន្នន័យមុនពេលដែលអ្នកអាចប្រើប្រាស់វា។ នេះធានាឱ្យមានសុវត្ថិភាពនៃប្រភេទទិន្នន័យ និងជួយការពារកំហុសដែលអាចកើតឡើង។

- ប្រភេទទិន្នន័យក្នុង Java

ប្រភេទទិន្នន័យក្នុង Java ចែកចេញជាប្រភេទចម្បង ២:

+ Primitive Data Types (ប្រភេទទិន្នន័យគោល):

គឺជាប្រភេទទិន្នន័យមូលដ្ឋានដែលបានផ្តល់ឱ្យដោយភាសា Java។ Java មានប្រភេទទិន្នន័យគោលចំនួន ៨:

- byte: អគ្គលេខ 8-bit, មានជួរតម្លៃពី -128 ដល់ 127។
- short: អគ្គលេខ 16-bit, មានជួរតម្លៃពី -32,768 ដល់ 32,767។
- int: អគ្គលេខ 32-bit, មានជួរតម្លៃពី -2^31 ដល់ 2^31-1។
- long: អគ្គលេខ 64-bit, មានជួរតម្លៃពី -2^63 ដល់ 2^63-1។
- float: លេខទសភាគ 32-bit, ដែលប្រើសម្រាប់តម្លៃទសភាគសកល។

- double: លេខទសភាគ 64-bit, ដែលប្រើសម្រាប់តម្លៃទសភាគសតួ។
- char: តួអក្សរ 16-bit Unicode។
- boolean: ជាតម្លៃនៃ 2 ប្រភេទ true និង `false`។

តារាងទី១. Java Primitive Data Type.

Data Type	Description	Size	Default Value
boolean	true or false	1-bit	false
char	Unicode Charact	16-bit	\u0000
byte	Signed Integer	8-bit	(byte) 0
Short	Signed Integer	16-bit	(short) 0
Int	Signed Integer	32-bit	0
Long	Signed Integer	64-bit	0L
float	Real number	32-bit	0.0f
double	Real number	64-bit	0.0d

តារាងទី២. Java floating-point

Type	Size		Range	Precision
name	bytes	bits	approximate	in decimal digits
float	4	32	+/- 3.4 * 1038	6-7
double	8	64	+/- 1.8 * 10308	15

តារាងទី៣. Key words in Java

abstract	continue	For	new
switch	assert	default	goto
package	synchronized	boolean	do
if	private	this	break
double	implements	protected	throw
byte	else	import	public
throws	case	enum	instanceof
return	transient	catch	extends
int	short	try	char
final	interface	static	void
class	finally	long	
volatile	const	float	native
super	while		

+. Reference Data Types (ប្រភេទទិន្នន័យយោង):

គឺជាវត្ថុ (objects) ដែលអាចប្រើដើម្បីចូលដំណើរការវត្ថុផ្សេងៗ។ វាមានរួមបញ្ចូល:

- Classes (ថ្នាក់): ដូចជា String, Scanner, ឬថ្នាក់ផ្ទាល់ខ្លួន។

- Interfaces (ចំណុចប្រទាក់): ប្រើដើម្បីកំណត់មុខងារដែលត្រូវអនុវត្តដោយថ្នាក់ផ្សេងៗ។

- Arrays (អារេ): ជាវត្ថុដែលអាចទុកតម្លៃរាប់ចំនួនដែលមានប្រភេទទិន្នន័យតែមួយ។

- Enums (លំដាប់ស្ថានភាព): ថ្នាក់ពិសេសដែលតំណាងឱ្យក្រុមនៃតម្លៃថេរ។

ការយល់ដឹងពីប្រភេទទិន្នន័យគឺសំខាន់ក្នុង Java ព្រោះវាមានឥទ្ធិពលលើការប្រើប្រាស់អង្គចងចាំ និងលើប្រតិបត្តិការណ៍ដែលអាចត្រូវបានធ្វើលើអថេរនីមួយៗ។

- Data Type របស់ C#

Data type នៅក្នុងភាសា C# គឺជាប្រភេទទិន្នន័យដែលកំណត់ប្រភេទនៃតម្លៃដែលអាចបានទុកក្នុងអថេរមួយ។ នៅក្នុង C#, អថេរត្រូវតែប្រកាសជាមួយប្រភេទទិន្នន័យ មុនពេលអ្នកអាចប្រើប្រាស់វា។ C# គឺជាភាសាមានការកំណត់ប្រភេទច្បាស់លាស់ (statically typed language) ដូច្នេះប្រភេទទិន្នន័យមានសារៈសំខាន់ក្នុងការគ្រប់គ្រងរាល់ប្រភេទនៃទិន្នន័យ និងការប្រតិបត្តិអថេរនីមួយៗ។

- ប្រភេទទិន្នន័យក្នុង C#

C# មានប្រភេទទិន្នន័យចម្បង ២ ប្រភេទដូចនេះ៖

+ Primitive Data Types (ប្រភេទទិន្នន័យគោល):

ទាំងនេះគឺជាប្រភេទទិន្នន័យដែលបានបង្កើតមកជាមួយភាសា C# ហើយមានដូចខាងក្រោម៖

- byte: អគ្គលេខ 8-bit, មានជួរតម្លៃពី 0 ដល់ 255។
- sbyte: អគ្គលេខ 8-bit, មានជួរតម្លៃពី -128 ដល់ 127។
- short: អគ្គលេខ 16-bit, មានជួរតម្លៃពី -32,768 ដល់ 32,767។
- ushort: អគ្គលេខ 16-bit, មានជួរតម្លៃពី 0 ដល់ 65,535។
- int: អគ្គលេខ 32-bit, មានជួរតម្លៃពី -2^31 ដល់ 2^31-1។
- uint: អគ្គលេខ 32-bit, មានជួរតម្លៃពី 0 ដល់ 4,294,967,295។
- long: អគ្គលេខ 64-bit, មានជួរតម្លៃពី -2^63 ដល់ 2^63-1។
- ulong: អគ្គលេខ 64-bit, មានជួរតម្លៃពី 0 ដល់ 18,446,744,073,709,551,615។
- float: លេខទសភាគ 32-bit, ប្រើសម្រាប់តម្លៃទសភាគសកល។
- double: លេខទសភាគ 64-bit, ប្រើសម្រាប់តម្លៃទសភាគសតួ។
- char: តួអក្សរ 16-bit Unicode។
- bool: តម្លៃនៃ 2 ប្រភេទ true និង `false`។

តារាងទី៤. C# Primitive Data Types

C# type name	.NET type name	Description
`bool`	`System.Boolean`	Represents a Boolean value. It can be set to `true` or `false`
`byte`	`System.Byte`	Represents an 8-bit unsigned integer
`char`	`System.Char`	Represents a UTF-16 code unit
`decimal`	`System.Decimal`	128-bit data type suitable for financial calculations
`double`	`System.Double`	Double-precision floating-point value
`float`	`System.Single`	Single precision floating-point value
`int`	`System.Int32`	Represents a 32-bit integer value
`long`	`System.Int64`	Represents a 64-bit integer value
`sbyte`	`System.SByte`	Represents an 8-bit signed integer
`short`	`System.Int16`	Represents a 16-bit signed integer
`uint`	`System.UInt32`	Represents a 32-bit unsigned integer
`ulong`	`System.UInt64`	Represents a 64-bit unsigned integer
`ushort`	`System.UInt16`	Represents a 16-bit unsigned integer

តារាងទី៥. Float type និង Decimal type

Type	Size (in bits)	precision	Range
float	32	7 digits	1.5 x 10-45 to 3.4 x 1038
double	64	15-16 digits	5.0 x 10-324 to 1.7 x 10308
decimal	128	28-29 decimal places	1.0 x 10-28 to 7.9 x 1028

តារាងទី៦. Key words C#

params	private	protected	public	readonly
ref	Return	sbyte	sealed	short
sizeof	stackalloc	static	string	struct
switch	this	throw	true	thy
typeof	uint	ulong	unchecked	void
volatile	while	char	checked	class
const	continue	decimal	default	delegate
do	double	else	enum	event
explicit	extern	false	finally	fixed
floot	for	foreach	goto	if
implicit	in	In (generich modifier)	int	interface
internal	is	lock	long	namespace
null	new	object	operator	out
Out (generich modifier)	override	abstract	as	base
bool	break	byte	case	catch
unsafe	oshort	using	Using static	

+ Reference Data Types (ប្រភេទទិន្នន័យយោង):

ប្រភេទទិន្នន័យនេះផ្តល់ឱ្យ C# អាចទៅយោងទៅនឹងវត្ថុជាក់លាក់មួយ ។ វាមានរួមបញ្ចូល៖

- Classes (ថ្នាក់): e.g., String, List, custom classes.

- Interfaces (ចំណុចប្រទាក់): ប្រើសម្រាប់កំណត់មុខងារដែលត្រូវអនុវត្តដោយថ្នាក់ផ្សេងៗ។

- Arrays (អារេ): ជាវត្ថុដែលអាចទុកតម្លៃជាច្រើនដែលមានប្រភេទទិន្នន័យតែមួយ។

- Delegates: ប្រភេទទិន្នន័យដែលអនុញ្ញាតឱ្យអ្នកផ្ទេរព័ត៌មាននៃមុខងារមួយ។

ការយល់ដឹងពីប្រភេទទិន្នន័យនេះជាសារៈសំខាន់ក្នុងការសរសេរកូដប្រកបដោយប្រសិទ្ធភាព និងសុវត្ថិភាពក្នុងភាសា C#។

២.២. String Type

String នៅក្នុងភាសា Java និង C# គឺជាប្រភេទទិន្នន័យដែលប្រើសម្រាប់ការផ្ទុក និងគ្រប់គ្រងអក្សរឬចំនួនមួយនៃតួអក្សរ។ ក្នុងភាសាទាំងពីរ String គឺជា class (ថ្នាក់) ដែលមានមុខងារនិងវិធីសាស្ត្រដើម្បីគ្រប់គ្រងអក្សរ។ តែប៉ុន្តែ String នៅក្នុង Java និង C# មានខុសស្លែកគ្នាផ្នែកខ្លះទៀតផង។

+String នៅក្នុង Java

- Class String នៅក្នុង Java ត្រូវបានបង្កើតជាអ្វីមួយដែលតំណាងឱ្យអត្ថបទមួយ។

- String នៅក្នុង Java គឺ immutable ដែលមានន័យថា ពេលដែល String ត្រូវបានបង្កើតឡើង មិនអាចប្តូរតម្លៃក្នុងអ្វីដែលវាគ្រប់គ្រងបានទេ។ ប្រសិនបើអ្នកចង់ផ្លាស់ប្តូរតម្លៃវា Java នឹងបង្កើត String ថ្មីមួយ។

- String នៅក្នុង Java ត្រូវបានគ្រប់គ្រងដោយអង្គចងចាំដែលហៅថា String pool ដែលជាកន្លែងដែល Java គ្រប់គ្រង String ដើម្បីប្រើសម្រាប់ធនធាន។

+ String នៅក្នុង C#

- Class String នៅក្នុង C# គឺដូចគ្នា Java ផងដែរដែលតំណាងឱ្យអត្ថបទមួយ និងទ្រទ្រង់មុខងារច្រើនសម្រាប់គ្រប់គ្រងអក្សរ។

- ដូចជា Java, String នៅក្នុង C# ក៏ជា immutable ដែរ។ អក្សរជាធម្មតាធ្វើអោយមាន String ថ្មីមួយត្រូវបានបង្កើតឡើង។

- C# ក៏មានការគ្រប់គ្រងអង្គចងចាំដូច Java តែវាមិនបានជាក់លាក់ដូច Java។

+ភាពខុសគ្នាចម្បង

- Syntax: ប្រភេទទិន្នន័យ String ត្រូវបានសរសេរទាំងជាទ្រង់ទ្រាយតូច (lowercase) "string" ក្នុង C# សម្រាប់ keyword, ខណៈពេលដែល Java គឺត្រូវបានសរសេរទាំងជាទ្រង់ទ្រាយធំ (uppercase) "String".

- String Pool: Java មាន String pool ដែលជាកន្លែងពិសេសក្នុងអង្គចងចាំសម្រាប់គ្រប់គ្រង String សម្រាប់ធនធាន។ នៅ C# វាមានការគ្រប់គ្រងជាទូទៅ តែវាមិនបានច្បាស់លាស់ដូច Java។

- Performance: ការគ្រប់គ្រងអង្គចងចាំទាំង String pool នៅក្នុង Java អាចធ្វើឱ្យមានប្រសិទ្ធិភាពខ្ពស់ជាង C# ក្នុងករណីមួយចំនួនពាក់ព័ន្ធនឹង String.

ទោះជាយ៉ាងណា, String ក្នុង Java និង C# គឺមានសមត្ថភាពនិងមុខងារជាធម្មតាជាច្រើនដែលស្រដៀងគ្នា។

២.៣ . Struct

Struct នៅក្នុងភាសា Java និង C# គឺជាប្រភេទទិន្នន័យមួយដែលត្រូវបានប្រើប្រាស់ដើម្បីរក្សាតម្លៃដែលទាក់ទងគ្នាក្នុង context ឬវិធីសាស្ត្រមួយ។ Struct គឺជាប្រភេទទិន្នន័យដែលមានលក្ខណៈផ្ទុក Value Types ហើយសាមញ្ញជាង Classes។

+Struct នៅក្នុង C#

- Struct គឺជាប្រភេទទិន្នន័យដែលអាចប្រកាសដោយប្រើពាក្យ struct ។

- Struct គឺជា Value Type ដែលមានន័យថា នៅពេលអ្នកចម្លង struct មួយទៅអថេរមួយទៀត វាត្រូវបានចម្លងតម្លៃទាំងមូល។

- Struct ត្រូវបានប្រើជាទូទៅសម្រាប់ទិន្នន័យតូច និងកំរិតនៃការអនុញ្ញាតជាក់លាក់ (Immutable) ដែលមិនចាំបាច់គ្រប់គ្រងនៅក្នុង heap memory។

- Struct អាចមាន fields, properties, methods, constructors និង interfaces ប៉ុន្តែមិនអាចមាន inheritance ឬ destructors ដូច classes។

+Struct នៅក្នុង Java

- នៅក្នុង Java Struct មិនមានឧបករណ៍ដាក់ក្នុងភាសាដូច C# ទេ។ Struct ត្រូវបានជំនួសដោយ Classes និង * *Primitive Types* *។

- Java ផ្តល់នូវប្រភេទទិន្នន័យដែលជា primitive types ដូចជា int, float, និង classes ដើម្បីផ្ទុក និងគ្រប់គ្រងទិន្នន័យដែលមានលក្ខណៈប្រព័ន្ធដូច struct ក្នុង C#។

- ដើម្បីរក្សាចំណេះដឹងទាំងអស់នេះនៅក្នុង Java អ្នកអាចប្រើ Classes ជំនួសដើម្បីកំណត់នូវទិន្នន័យដែលទាក់ទងគ្នានេះ។

+ភាពខុសគ្នាចម្បង

- ជនជាតិក្នុងភាសា: C# មាន Struct ជាឧបករណ៍ក្នុងភាសាដែល Java មិនមាន។ Java ប្រើ classes ទាំងស្រុងនិង primitive types ឬ wrapper classes ដើម្បីបំពេញមុខងារដូច struct នៅ C#។

- ប្រភេទទិន្នន័យ: Struct នៅ C# ជា Value Type ដែលមានន័យថា វាត្រូវបានផ្ទុកនៅក្នុង stack memory ប៉ុន្តែ Classes នៅ Java និង C# ជា Reference Type ដែលផ្ទុកតម្លៃនៅក្នុង heap memory។

- Inheritance: Struct នៅ C# មិនគាំទ្រ inheritance នោះមានន័យថា Struct មិនអាចបង្កើតដោយប្រើ base struct ឬ class ហើយក៏មិនអាចបង្កើត sub-struct បានដែរ។ Classes នៅ Java និង C# អាចមាន inheritance និងបង្កើត classes ឬ subclasses ផ្សេងទៀតបាន។

- Immutable និង class : Structs ត្រូវបានប្រើជាទូទៅសម្រាប់ទិន្នន័យដែលមិនមានការផ្លាស់ប្តូរ (immutable) ហើយ classes ត្រូវបានប្រើសម្រាប់ទិន្នន័យដ៏ស្មុគស្មាញដែលត្រូវការសមត្ថភាពពេញលេញ។

២.៤ Inherence

Inheritance នៅក្នុងភាសា Java និង C# គឺជាមុខងារមួយក្នុងកម្មវិធីរចនាក្នុងការអភិវឌ្ឍន៍កម្មវិធី ដែលអនុញ្ញាតឱ្យថ្នាក់ (class) មួយអាចទទួលបន្សំមុខងារពីថ្នាក់មួយផ្សេងទៀត។ វាផ្តល់នូវសមត្ថភាពឱ្យអ្នកអភិវឌ្ឍន៍អាចបង្កើតថ្នាក់ថ្មីដែលមានសមត្ថភាពឬវិធីសាស្ត្ររបស់ថ្នាក់ដែលមានរួចហើយ (parent class) ហើយក៏អាចបន្ថែមលក្ខណៈពិសេសថ្មីៗមួយចំនួនផងដែរ។

+Inheritance នៅក្នុង Java

- នៅក្នុង Java, inheritance អនុញ្ញាតឱ្យ class មួយអាច extends (ពង្រីក) មកពី class មួយផ្សេងទៀត។

- ការដាក់ពាក្យ extends គឺសម្រាប់ការប្រកាសថ្នាក់ថ្មីដែលទទួលបន្សំពីថ្នាក់ផ្សេងទៀត។

- Java គាំទ្រ single inheritance (ការទទួលបន្សំត្រឹមតែមួយថ្នាក់ប៉ុណ្ណោះ) ប៉ុន្តែអាចរួមបញ្ចូល multiple inheritance តាមរយៈ interfaces។

ក្នុងឧទាហរណ៍នេះ, Dog class ទទួលបន្សំពី Animal class ហើយអាចប្រើវិធីសាស្ត្រ eat( ) របស់ Animal ។

+Inheritance នៅក្នុង C#

- នៅក្នុង C#, inheritance អនុញ្ញាតឱ្យ class មួយអាច inherits ពី class មួយផ្សេងទៀតតាមរយៈពាក្យ (colon)។

- C# ក៏គាំទ្រ single inheritance ដូច Java តែគាំទ្រ multiple inheritance តាមរយៈ interfaces។

នៅក្នុងឧទាហរណ៍នេះ, Dog class ទទួលបន្ទុំពី Animal class ហើយអាចប្រើវិធីសាស្ត្រ Eat( ) របស់ Animal ។

+ភាពខុសគ្នាចម្បង

- Syntax: Java ប្រើពាក្យ extends ក្នុងករណី inheritance ខណៈពេល C# ប្រើ * *: (colon) * *។

- Multiple Inheritance: Java និង C# មិនគាំទ្រ multiple inheritance ទៅលើ classes ទេ តែសម្រាប់ interfaces វាអាចធ្វើបានទាំងពីរ។

- Access Modifiers: C# ផ្តល់នូវ access modifiers បន្ថែមដូចជា protected internal ដែលធ្វើឱ្យវាត្រួសត្រាយបន្ថែមក្នុងការគ្រប់គ្រង inheritance។

ទោះជាយ៉ាងណា, Inheritance ក្នុង Java និង C# មានសមត្ថភាពស្រដៀងគ្នា ហើយមានមូលដ្ឋានគ្រឹះដូចគ្នានៃការអភិវឌ្ឍកម្មវិធី។

២.៥ . Array

Array នៅក្នុងភាសា Java និង C# គឺជាប្រភេទទិន្នន័យដែលអនុញ្ញាតឱ្យអ្នកតួរអាចទុកតម្លៃមួយចំនួនដែលមានប្រភេទទិន្នន័យដូចគ្នាក្នុងចំណាត់ថ្នាក់តែមួយ។ Array មានមុខងារចាំបាច់ក្នុងការចងក្រង និងគ្រប់គ្រងទិន្នន័យដែលមានទំហំដូចគ្នា។

+Array នៅក្នុង Java

- នៅក្នុង Java, Array គឺជា object ដែលអាចទុកតម្លៃច្រើនរបស់ប្រភេទទិន្នន័យតែមួយ។

- ប្រសិនបើតម្លៃរបស់ Array ត្រូវបានប្រកាសនៅពេលសរសេរកូដ, ទំហំនៃ Array នឹងមិនអាចផ្លាស់ប្តូរបានទេ (fixed size)។

- Java អាចត្រូវបានបង្កើតជា one-dimensional ឬ multi-dimensional arrays (ដូចជា អារេទ្វេដង់ស្យុង)។

- Java ផ្តល់នូវ Array indexing ចាប់ពី 0 ដែលសំគាល់តម្លៃក្នុងកន្លែងតែមួយមួយរបស់ Array ។

- នៅក្នុង C#, Array គឺជា object ដែលអាចទុកតម្លៃច្រើននៃប្រភេទទិន្នន័យតែមួយដែរ។

- Array នៅក្នុង C# ក៏មានទំហំដែលត្រូវបានកំណត់ច្បាស់លាស់ ដូច Java។

- C# អាចបង្កើត single-dimensional, multi-dimensional, និង jagged arrays (អារេដែលអាចមានទំហំផ្សេងៗ)។

- Array នៅ C# មាន index ដែលចាប់ពី 0 ដូច Java ។

+ភាពខុសគ្នាចម្បង

- Syntax: វិធីសាស្ត្រនៃការបង្កើត និងប្រើប្រាស់ Array សម្រាប់ Java និង C# មានសាមញ្ញដូចគ្នា ប៉ុន្តែ syntax របស់វាមានភាពខុសគ្នា ដោយពាក្យដាក់ចុង ; ក្នុង Java និង [] នៅទាំង Java និង C# នេះដូចគ្នា។

- Initialization: Java អាចបង្កើត និងផ្តល់តម្លៃផ្តាច់មុខ (initialize) Array ក្នុងមួយឃ្លាដោយប្រើ {} ខណៈពេល C# អាចធ្វើដូចគ្នាឬអាចបង្កើត array មួយគ្មានតម្លៃឱ្យតែបង្កើតមុនហើយដាក់តម្លៃក្រោយ។

- Multi-dimensional Array: Java គាំទ្រ multi-dimensional arrays តាមរយៈ syntax [][], ខណៈ C# ផ្តល់ការគាំទ្រ multi-dimensional arrays ដោយប្រើសញ្ញាកំណត់ [,] និងត្រូវបានប្រើជាទូទៅក្នុង context.

- Array Class Methods: C# ផ្តល់នូវក្រុមមុខងារដូចជា Array.Sort( ), Array.Reverse( ) ដែលអាចប្រើជាមធ្យោបាយស្រួលក្នុងការគ្រប់គ្រង និងផ្លាស់ប្តូរទិន្នន័យ Array ប៉ុន្តែ Java ប្រើ Arrays utility class សម្រាប់មុខងារដូចគ្នា។

ក្នុងសមាមាត្រដូចគ្នា, Array នៅ Java និង C# ធ្វើការបង្ហាញ និងដំណើរការដូចគ្នាប៉ុន្តែត្រូវបានអភិវឌ្ឍន៍ដោយខុសគ្នានៃ syntax និង context អង្គតូចៗក្នុងភាសាទាំងពីរ។

២.៦ Reference and Value Type

Reference Type និង Value Type នៅក្នុងភាសា Java និង C# គឺជាប្រភេទទិន្នន័យ ដែលមានលក្ខណៈផ្សេងគ្នាក្នុងរបៀបផ្ទុក និងដំណើរការទិន្នន័យ។ មុនពេលពិភាក្សាពីភាពខុសគ្នារវាង Java និង C#, យើងត្រូវមើលយល់ពីចំនុចមូលដ្ឋាននៃ Reference Type និង Value Type ជាមុនសិន។

+Value Type

- Value Type គឺជាប្រភេទទិន្នន័យដែលតម្លៃរបស់វាត្រូវបានផ្ទុកនៅក្នុងអង្គចងចាំដោយផ្ទាល់។ នោះមានន័យថា នៅពេលអ្នកកំណត់ Value Type ឱ្យអថេរមួយ វាត្រូវបានផ្ទុកតម្លៃក្នុងអថេរនោះផ្ទាល់។

- នៅពេលអ្នកចម្លង Value Type ទៅអថេរមួយទៀត វាត្រូវបានចម្លងតម្លៃទាំងមូលឆ្ពោះទៅកាន់អថេរថ្មីនោះដោយផ្ទាល់។

- នៅ Java និង C#, Value Types ដូចជា primitive types (int, char, bool, etc.) និង structs (សម្រាប់ C#) ត្រូវបានរាប់បញ្ចូលក្នុងប្រភេទទិន្នន័យនេះ។

+Reference Type

- Reference Type គឺជាប្រភេទទិន្នន័យដែលមិនផ្ទុកតម្លៃតាមផ្ទាល់ទេ ប៉ុន្តែផ្ទុកពត៌មានអំពីអង្គចងចាំដែលវាត្រូវបានផ្ទុក។ នោះមានន័យថា នៅពេលអ្នកកំណត់ Reference Type ឱ្យអថេរមួយ វាត្រូវបានផ្ទុកតែអាសយដ្ឋាន (memory address) ឬការយោងទៅរកតម្លៃនោះ។

- នៅពេលអ្នកចម្លង Reference Type ទៅអថេរមួយទៀត វាត្រូវចម្លងតែអាសយដ្ឋានដូចគ្នា។ នេះមានន័យថាអថេរទាំងអស់នឹងស្ថិតនៅលើតម្លៃតែមួយដែលស្ថិតនៅអង្គចងចាំនោះ។

- នៅ Java និង C#, Reference Types រួមបញ្ចូល String, classes, arrays និង objects.

+ ភាពខុសគ្នាចម្បងរវាង Java និង C#

- ការប្រកាស: នៅ Java និង C#, Value Types ត្រូវបានប្រកាសក្នុងគោលបំណងដូចគ្នា, តែ C# មាន additional struct data type ដែលគឺជាប្រភេទទិន្នន័យដែលរាប់បញ្ចូលនៅក្នុង Value Types។

- Memory Management: នៅ Java, Reference Types ត្រូវបានគ្រប់គ្រងដោយ JVM (Java Virtual Machine) ដោយប្រើ garbage collection system ដើម្បីសំអាត objects មិនមានអ្នកប្រើត្រឡប់។ នៅ C#, CLR (Common Language Runtime) ផ្តល់នូវការគ្រប់គ្រងអង្គចងចាំតាមរយៈ garbage collection ដូច Java ប៉ុន្តែមានការបន្ថែមលើ stack និង heap memory management ដែលបែងចែកច្បាស់លាស់នៃ Value Types និង Reference Types។

- Default Values: នៅ C#, Value Types ទាំងអស់មាន default values (e.g., int = 0, bool = false), ហើយ Reference Types មាន default value of null។ Java មិនមាន default values សម្រាប់ primitive types ដែលត្រូវបានប្រកាសនៅក្រៅនៃ context, ការកំណត់តម្លៃមុន (initialization) គឺចាំបាច់។

- Boxing and Unboxing: នៅ C#, បន្ទាប់ពីអ្នកចង់ផ្លាស់ប្តូរទិន្នន័យពី Value Type ទៅ Reference Type ឬ វិញគ្នា, វាគឺជាប្រព័ន្ធមួយដែលគេស្គាល់ថា boxing និង unboxing. Java គឺមាន autoboxing, ដែលជាការប្តូរ ជាអចិន្ត្រៃយ៍ពី primitive types ទៅ Wrapper Classes (Integer, Character, etc.)។

- Simplicity of Usage: នៅ Java, String ដែលជាមួយនឹង Reference Types ត្រូវបានគ្រប់គ្រងយ៉ាងងាយ ស្រួលដោយ JVM. C# អាចប្រើបានងាយតែមានលក្ខណៈសំខាន់សម្រាប់ប្រភេទទិន្នន័យដែលត្រូវបានគ្រប់គ្រង ក្នុង context ផ្សេងៗ។

ក្នុងសន្និដ្ឋាន, Reference Types និង Value Types មានលក្ខណៈសម្បទានសំខាន់ជាដូចគ្នា ទោះបីជា Java និង C# មានប្រព័ន្ធគ្រប់គ្រងដែលមានលក្ខណៈត្រួសត្រាយខុសគ្នាក៏ដោយ។

២.៧. Pointer

Pointer នៅក្នុងភាសា Java និង C# គឺជាប្រធានបទដែលមានការគ្រប់គ្រង និងការប្រើប្រាស់ខុសគ្នាបន្តិច។ យើងនឹងពិភាក្សាពី pointer ក្នុងភាសាទាំងពីរនេះ ហើយបង្ហាញពីភាពខុសគ្នារវាងវា។

+ Pointer នៅក្នុងភាសា C#

- Pointer គឺជាប្រភេទអថេរ ដែលរក្សាទុកអាសយដ្ឋានអង្គចងចាំ (memory address) របស់តម្លៃ។ វា អនុញ្ញាតឱ្យអ្នកអភិវឌ្ឍន៍អាចចូលដំណើរការទៅកាន់ memory location ដោយផ្ទាល់។

- នៅក្នុង C#, pointers ត្រូវបានគ្រប់គ្រងនៅក្នុង context ឬ block ដែលត្រូវបានស្គាល់ថា unsafe (មិនមាន សុវត្ថិភាព) ហើយការប្រើប្រាស់ pointers ត្រូវការការអនុញ្ញាតពិសេស។

- Pointers ជាទូទៅត្រូវបានប្រើក្នុងកម្មវិធីដែលទាមទារការគ្រប់គ្រងអង្គចងចាំកម្រិតទាប ឬក្នុង context ដែលតម្រូវឱ្យមានប្រសិទ្ធភាពខ្ពស់។

+Pointer នៅក្នុងភាសា Java

- Java មិនគាំទ្រដល់ pointer ដោយផ្ទាល់ទេ។ ហេតុនេះ វាមិនអនុញ្ញាតឱ្យអ្នកអភិវឌ្ឍន៍ចូលដំណើរការទៅកាន់ memory address ដោយផ្ទាល់ដូច C# ទេ។

- កម្មវិធី Java គ្រប់គ្រង memory ដោយស្វ័យប្រវត្តិតាមរយៈ Java Virtual Machine (JVM) ហើយចូល ដំណើរការដោយស្វ័យប្រវត្តិដើម្បីបញ្ចេញ memory ដែលមិនមានការប្រើប្រាស់ (Garbage Collection)។

- ចំពោះកម្មវិធីដែលអាចប្រើ pointers, Java បានដកចេញនូវការគ្រប់គ្រងនេះ ដើម្បីជៀសវាងបញ្ហាដូច buffer overflow និងធានាសុវត្ថិភាព។

- ក្នុង Java, អ្នកអភិវឌ្ឍន៍ប្រើប្រាស់ references ដែលជាការស្វែងយល់ទៅកាន់ object ក្នុង heap memory ប៉ុន្តែមិនអាចចូលដំណើរការទៅកាន់ memory address ដោយផ្ទាល់នោះទេ។

+ភាពខុសគ្នាចម្បងរវាង Pointer នៅក្នុង Java និង C#

ការគាំទ្រផ្ទាល់:

- C# អនុញ្ញាតឱ្យប្រើ pointers តាមរយៈ unsafe code block។

- Java មិនគាំទ្រ pointers ហើយគ្រប់គ្រង memory ដោយស្វ័យប្រវត្តិដោយ JVM។

Memory Safety:

- C# អាចមានការប្រឈមមុខនឹងបញ្ហានៃ memory safety នៅពេលប្រើ pointers នៅក្នុង context ដែលមិនមានសុវត្ថិភាព (unsafe context) ប៉ុន្តែសម្រាប់កម្មវិធីសាកលវិទ្យាត្រូវការប្រើប្រាស់នូវតម្លៃដើម្បីប្រើប្រាស់អង្គចងចាំ។

- Java ជៀសវាងបញ្ហានេះដោយមិនអនុញ្ញាតឱ្យប្រើ pointers ហើយមានសុវត្ថិភាពដោយប្រើ garbage collection។

Flexibility and Control:

- C# ផ្តល់នូវភាពបត់បែនបន្ថែម ដោយអនុញ្ញាតឱ្យអ្នកអភិវឌ្ឍន៍គ្រប់គ្រង memory ដោយផ្ទាល់និងប្រើ pointers នៅក្នុង context តម្រូវ (unsafe code block)។

- Java ផ្តោតលើការងាយស្រួលប្រើ និងសុវត្ថិភាព ដោយដកចេញការគ្រប់គ្រង memory ដោយផ្ទាល់ និងប្រើ references ជំនួស pointers។

២.៨ Partial Classes

Partial Class គឺជាការរចនាដែលអនុញ្ញាតឱ្យអ្នកអភិវឌ្ឍន៍អាចបែងចែក class មួយឲ្យទៅជា sections ឬ files ច្រើន ហើយពួកវានឹងត្រូវបានបញ្ចូលជាមួយគ្នាពេលកូដត្រូវបានបំលែង។ Partial Class អាចជាមធ្យោបាយល្អក្នុងការចែកកូដទៅតាមមុខងារផ្សេងៗ ឬនៅពេលដែល class ធំពេញ។

+ Partial Class នៅក្នុង C#

- Partial Class គឺជា feature មួយនៅក្នុង C# ដែលអនុញ្ញាតឱ្យអ្នកអភិវឌ្ឍន៍អាចបែងចែក class ឬ struct ទៅក្នុងពាក្យពេចន៍ផ្សេងៗច្រើន files ម្តងមួយ។

- ដើម្បីប្រកាស partial class អ្នកប្រើពាក្យគន្លឹះ partial មុនពាក្យគន្លឹះ class ហើយត្រូវមាន class ដែលមានឈ្មោះដូចគ្នាទាំងអស់នៅក្នុង files ច្រើន។

- Partial classes ត្រូវបានប្រើប្រាស់ក្នុងការចែកកូដដែលធំខ្លាំងដើម្បីធ្វើឱ្យវាសាមញ្ញនិងងាយស្រួលក្នុងការគ្រប់គ្រង ជាពិសេសនៅពេលដែល team អភិវឌ្ឍន៍កំពុងធ្វើការក្នុងគម្រោងតែមួយ។

+ Partial Class នៅក្នុង Java

- Java មិនគាំទ្រ concept នៃ Partial Class ទេ។ ក្នុង Java, class ត្រូវបានកំណត់នៅក្នុង file មួយប៉ុណ្ណោះ ហើយឈ្មោះ file នឹងត្រូវត្រូវនឹងឈ្មោះ class។

- បើអ្នកចង់បែងចែក functionality នៅក្នុង Java, អ្នកអាចប្រើ classes ឬ interfaces បន្ថែម ដើម្បីចែកសមត្ថភាពនានាទៅក្នុង classes ផ្សេងៗ។

+ ភាពខុសគ្នាចម្បង

ការគាំទ្រផ្នែកភាសា:

- C# គាំទ្រការប្រើប្រាស់ Partial Classes ដែលអាចបែងចែក class មួយទៅ files ច្រើនដើម្បីសម្រួលការអភិវឌ្ឍន៍។

- Java មិនគាំទ្រ Partial Classes ទេ។ Class មួយនៅ Java ត្រូវត្រូវបានកំណត់នៅក្នុង file មួយប៉ុណ្ណោះ។

ការបែងចែក Functionality:

- នៅក្នុង C#, Partial Classes អាចសម្រួលដល់ការបែងចែក function និងគ្រប់គ្រងកូដ។

- នៅក្នុង Java, អ្នកអាចបែងចែក function ដោយប្រើ classes ឬ interfaces បន្ថែមទៀត ប៉ុន្តែមិនអាចមានការបង្កើត class នៃឈ្មោះដូចគ្នាច្រើន files ទេ។

Team Collaboration:

- Partial Classes នៅ C# អាចប្រើប្រាស់បានប្រសើរ ក្នុង context ដែលអ្នកអភិវឌ្ឍន៍ច្រើនធ្វើការលើ class តែមួយ។

- នៅ Java, ការប្រើប្រាស់ classes និង interfaces ផ្សេងៗអាចជាជម្រើសដ៏ល្អសម្រាប់ការចនាឌីជីថលក្នុងក្រុម។

២.៩ Compiler Technology

Compiler Technology នៅក្នុងភាសា Java និង C# គឺជាបច្ចេកវិទ្យាដែលបំលែងកូដប្រភព (source code) ទៅជាកូដដែលអ្នកប្រព័ន្ធអាចអានបាន (machine-readable code) ហើយអាចដំណើរការបានដោយកុំព្យូទ័រ។

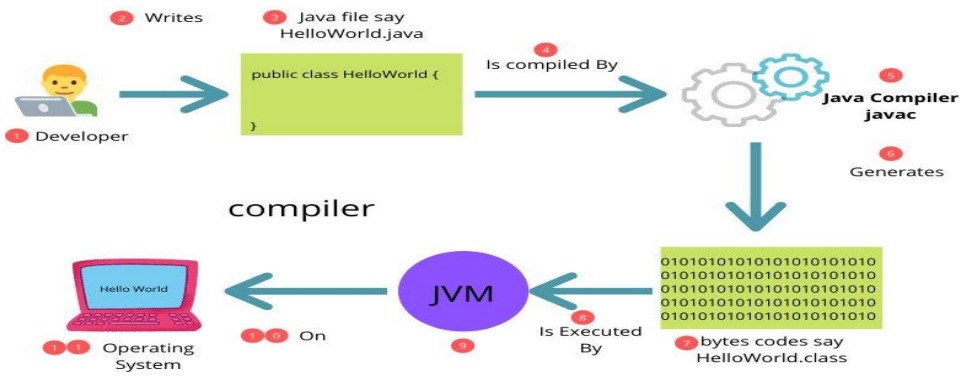
+ Java Compiler Technology

- Java Compiler (java) បំលែង Java source code ទៅជា bytecode ដែលជាប្រភេទ intermediary code (កូដពាក់កណ្តាល) ។

- Bytecode ត្រូវបានផ្ទុកនៅក្នុង .class files ហើយចុងក្រោយត្រូវបានដំណើរការដោយ Java Virtual Machine (JVM) ដែលអាចបញ្ជាក់វាទៅជា machine code សម្រាប់ប្រព័ន្ធដំណើរការនិមួយៗ។

- JVM គឺជាប្រព័ន្ធលើកទីពីរដែលអនុញ្ញាតឱ្យ Java ដំណើរការបាននៅលើប្រព័ន្ធដំណើរការជាច្រើន ដោយសារតែ JVM ត្រូវបានបង្កើតឡើងសម្រាប់មិនថាដំណើរការលើ Windows, Linux, MacOS និងផ្សេងៗទៀត។

រូបភាពទី១. Java compiler



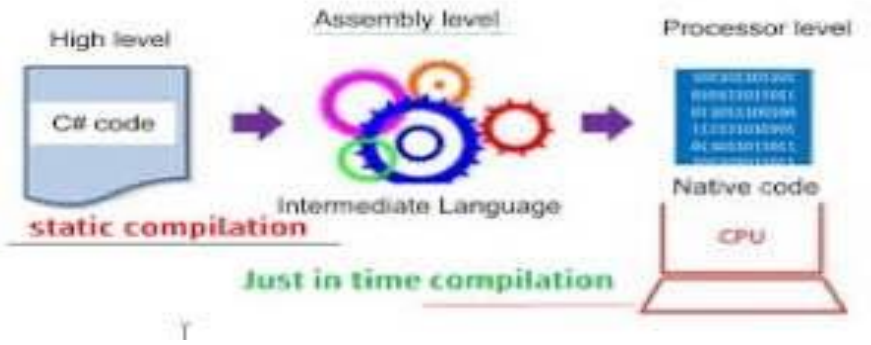
+ C# Compiler Technology

- C# Compiler (csc) បំលែង C# source code ទៅជា Intermediate Language (IL) ដែលជាប្រភេទ intermediary code ដូច Java bytecode។

- IL ត្រូវបានផ្ទុកនៅក្នុង .exe ឬ .dll files ហើយចុងក្រោយត្រូវបានដំណើរការដោយ Common Language Runtime (CLR) ដែលបញ្ជាក់វាទៅជា machine code សម្រាប់ Windows និងប្រព័ន្ធដំណើរការដែលត្រូវបានគាំទ្រ។

- CLR គឺជា runtime environment សម្រាប់ .NET applications ដែលអនុញ្ញាតឱ្យ C# និងភាសាផ្សេងៗក្នុង .NET ត្រូវបានបង្កើតលើ platform និមួយៗ។

រូបភាពទី២. C# compiler



+ ភាពខុសប្លែកចម្បង

បច្ចេកវិទ្យា និងវេទិកាដំណើរការ:

- Java ជា platform-independent ដែលមានន័យថា bytecode ត្រូវបានបង្ហាញដោយ JVM នៅលើវេទិកានិមួយៗដោយស្វ័យប្រវត្តិ។

- C# ត្រូវបានបង្កើតឡើងសម្រាប់ .NET Framework (ដំបូងគឺតែសម្រាប់ Windows) ប៉ុន្តែត្រូវបានពង្រីកទៅ .NET Core និង .NET 5/6/7 ដែលអាចដំណើរការបាននៅលើប្រព័ន្ធដំណើរការជាច្រើនដូចជា Linux និង MacOS។

គោលបំណងស្របច្បាប់ (Execution model):

- Java ចាំបាច់ត្រូវមាន JVM ដើម្បីដំណើរការដោយ Bytecode ត្រូវបានប្រើជាស្នូល និង JVM បញ្ជាក់ Bytecode ទៅជា machine code។

- C# ត្រូវការ CLR ដើម្បីដំណើរការបន្ទាប់ពី IL code ត្រូវបានបម្លែងជាដំណាក់កាល machine code ក្នុង context ពេលនោះ។

ប្រសិទ្ធភាពនៃការដំណើរការ:

- Java អាចមានបញ្ហាបន្តិចនៃប្រសិទ្ធភាពដោយសារតែ JVM ត្រូវការពេលក្នុងការបញ្ចូល និងបកប្រែ Bytecode ក្នុង runtime។

- C# អាចមានប្រសិទ្ធភាពខ្ពស់បន្ថែមដោយសារតែ JIT (Just-In-Time) compiler ដែលធ្វើការបកប្រែ IL code ទៅជា native code ពេលដែលដំណើរការត្រូវបានអនុញ្ញាត។

Tooling and Integration:

- Java អាចដំណើរការនៅលើប្រព័ន្ធដំណើរការនិមួយៗដោយ JVM បានត្រៀមសម្រាប់វេទិកាជាច្រើន។

- C# មានសិទ្ធិអនុញ្ញាតពិសេសក្នុង .NET ecosystem ដែលមាន Visual Studio ជាស្ថានីយការងារសម្រាប់ IDE និងមាន tooling សម្រាប់ការអភិវឌ្ឍន៍។

ជំពូកទី៣

ការអនុវត្តនិងប្រសិទ្ធភាព

៣.១ Object

Object នៅក្នុងភាសា Java និង C# គឺជាការចំណាយដ៏សំខាន់ដែលបង្កើតមូលដ្ឋានសម្រាប់ការអភិវឌ្ឍន៍ ហានាហាន:នៃ OOP (Object-Oriented Programming) ទាំងស្រុង។ Object ជាការច្នៃប្រឌិតនៃ Class និង ប្រើប្រាស់ដើម្បីផ្ទុកទិន្នន័យ និងចូលដំណើរការទៅនឹងវិធីសាស្ត្រនៅក្នុង Class ។

+Object នៅក្នុង Java

- Object នៅក្នុង Java គឺជា instance (ករណី) មួយនៃ Class ។

- Java គ្រប់គ្រង Object នៅក្នុង heap memory ។

- Java អនុញ្ញាតឱ្យបង្កើត Object ដោយប្រើ keyword new ហើយផ្តល់ឱ្យអ្នកអភិវឌ្ឍន៍នូវវិធីសាស្ត្រដែលអាច ចូលដំណើរការទៅនឹង attributes និង methods របស់ Class ។

- Java មាន Class មួយដែលស្គាល់ថា `Object` ដែលជាវិភាគទាននៃក្រុមទាំងអស់, និយាយទាំងសារភាពថា Class ទាំងអស់នឹងអាចសម្បូរទៅកាន់ Object ។

+Object នៅក្នុង C#

- Object នៅក្នុង C# គឺជា instance មួយនៃ Class ។

- C# គ្រប់គ្រង Object នៅក្នុង heap memory ដូច Java ។

- C# មាន keyword new សម្រាប់បង្កើត Object ។

- Class `Object` នៅ C# គឺជាក្រុមកំណត់បាច់របស់ប្រភេទទាំងអស់ក្នុង C# ហើយផ្តល់នូវ methods ជំនួយ សម្រាប់កម្មវិធី object-oriented ។

+ ភាពខុសគ្នាចម្បងរវាង Object នៅ Java និង C#

- . Inheritance Hierarchy:

- Java និង C# ទាំងពីរនឹងមាន Class `Object` ជាក្រុមកំណត់បាច់របស់ Class ទាំងអស់ ប៉ុន្តែមួយចំនួននៃ methods ដែលអាចប្រើបានគឺខុសគ្នា។

- Java's Object class មាន methods ដូចជា clone( ), finalize( ), notify( ), wait( ) ដែល C# មិនមានតាម default, ប៉ុន្តែ C# មាន methods ដូចជា GetType( ), MemberwiseClone( ) ជាដើម។

- . Memory Management:

- Java គ្រប់គ្រង memory តាមរយៈ Garbage Collection ដោយស្វ័យប្រវត្តិ។ Java អនុញ្ញាតឱ្យប្រើ threads ក្នុង JVM ដើម្បីស្វែងរកនិងសំអាត Object ដែលមិនបានប្រើទៀត។

- C# ក៏ប្រើប្រាស់ Garbage Collection ដោយស្វ័យប្រវត្តិផងដែរ តែអាចមានការកំណត់ដើម្បីកំណត់ threading pool ឬទៅដល់ការខំប្រាប់ garbage collector ដើម្បីសំអាត memory ក្នុង context ពិសេសមួយ។

- . Method Overloading and Overriding:

- Java និង C# មានរបៀបខុសគ្នាខ្លះនៃការ overriding methods, ជាពិសេសនៅក្នុង handling of exceptions (checked exceptions នៅ Java) និង access modifiers (protected/internal នៅ C#)។

- C# មាន `virtual` និង `override` keywords សម្រាប់ method overriding, ខណៈដែល Java ប្រើ `@Override` annotation។

- . Advanced Features:

- C# ផ្តល់នូវ features ដូចជា Properties (getter/setter) និង Delegates ដែលគ្មាននៅក្នុង Java, ទោះបីជា Java កំពុងមាន features ថ្មីៗនៅក្នុងរើសិនថ្មីៗក៏ដោយ។

៣.២ Test System

Test System នៅក្នុងភាសា Java និង C# ជាបច្ចេកវិទ្យាសម្រាប់ធ្វើសាកល្បងនិងធានានូវគុណភាពនៃកម្មវិធី។ យើងអាចប្រើ frameworks និង libraries ដូច nhauដើម្បីចាត់ការជាមួយ unit testing, integration testing, និងទិន្នន័យពាក់ព័ន្ធផ្សេងៗ។

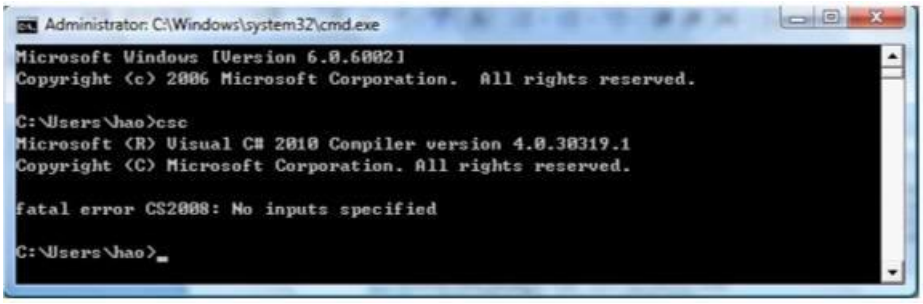
ចំណុចសំខាន់នៃការធ្វើតេស្ត

- Processor: Intel(R) Core(TM)2 Duo CPU P7350 @ 2.00GHZ
- Memory: 3072 MB DDR2 800MHZ(PC6400)
- Hard Disk: TOSHIBA mk3252GSX 320G
- Chipset: Intel Mobile 4 Series Chipset
- Operating System: Microsoft Windows Vista Home Premium (SP2)

+ C# compiler version

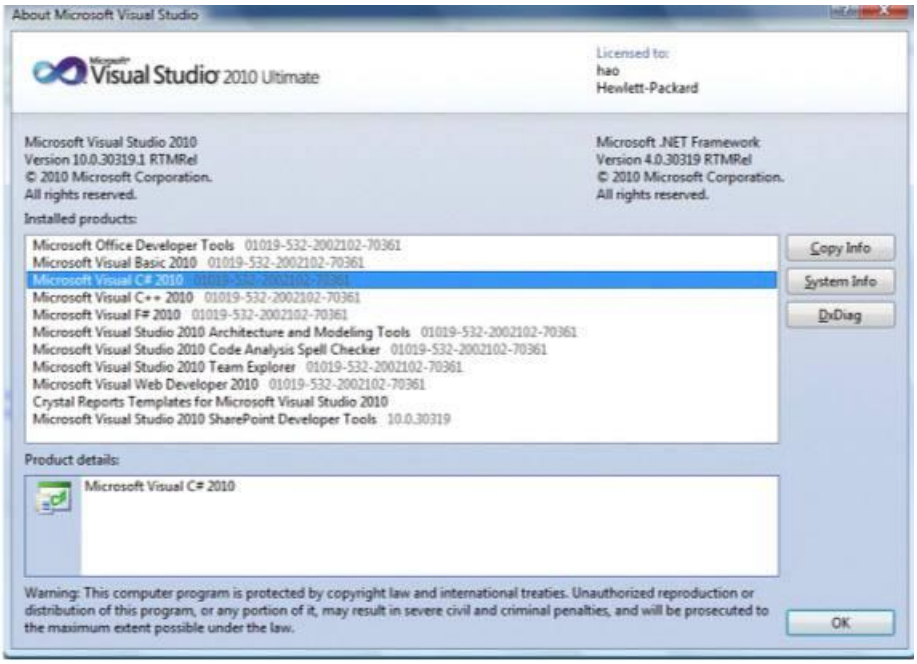
- Microsoft (R) Visual C# 2010 Compiler version 4.0.30319.1

រូបភាពទី៣.បង្ហាញ C# compiler version



- Microsoft (R) .NET Framework version 4.0.30319 RTMRel

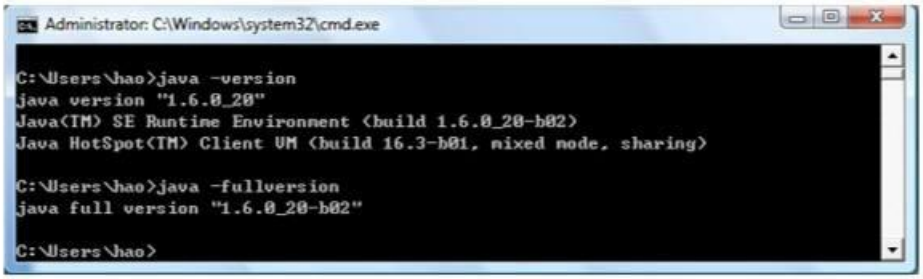
រូបភាពទី៤បង្ហាញពី Microsoft (R) .NET Framework version.



+ Java Version:

- Java full version "1.6.0_20-b02"

រូបភាពទី៥.បង្ហាញពី Java full version "1.6.0_20-b02"



- ភាពខុសគ្នាចម្បងរវាង Java និង C#

- . Testing Frameworks:

- Java មាន JUnit និង TestNG ដែលពេញនិយមបំផុតនិងមានភាពងាយស្រួលក្នុងការប្រើប្រាស់ testing frameworks ។

- C# មាន MSTest, NUnit, និង xUnit ដែលមានច្រើនលក្ខណៈពិសេសនិងសម្រួលដល់កំណត់ពិសេសនៃ context .NET ។

- . Tooling Integration:

- Java ការប្រើប្រាស់ testing frameworks ពេញនិយមនៅក្នុង IDEs ដូចជា IntelliJ, Eclipse, និង NetBeans ។

- C# មានការគាំទ្រដែលអាចធ្វើការជាមួយ Visual Studio, ដែលផ្តល់ការរួមបញ្ចូល testing engine និងចំណាត់ការមិនទាន់បានស្តាបនាតាម context សម្រាប់ .NET ។

- . Dependency Injection and Mocking:

- Java ប្រើប្រាស់ Mockito និង JMock សម្រាប់ mocking frameworks ។

- C# ប្រើប្រាស់ Moq និង FakeItEasy ជាឧបករណ៍សម្រាប់ mocking ។

- . Parallel Execution:

- TestNG នៅ Java មាន support សម្រាប់ parallel execution, នៅពេលដែល MSTest និង NUnit នៅ C# គាំទ្រដោយផ្នាល់នូវ features ដូច្នេះ។

- ទាំងពីរភាសានេះទាំងអស់កំពុងបំពេញតាមតម្រូវការសម្រាប់ធានាភាពទន់ភ្លន់និងការពេញចិត្តនៃកម្មវិធី។

៣.៣ Implementation

- Primitive Types: Java និង C# ស្មើរតែដូចគ្នាទាំងក្នុងការប្រើប្រាស់ int, double, និង long, ប៉ុន្តែការប្រតិបត្តិការផ្សេងគ្នាតាមប្រព័ន្ធ runtime (JVM)សម្រាប់ Java និង CLR សម្រាប់ C#)។

- Trig Functions: ទាំង Java និង C# ប្រើ Math library ដើម្បីគណនាលេខ angle, ប៉ុន្តែ Java បំលែង angle ទៅជា radians ជាស្វ័យប្រវត្តិដោយប្រើ Math.toRadians( ) នៅក្នុងប្រតិបត្តិការ trigonometry។
- I/O Operations: Java ត្រូវការទាញយក BufferedReader សម្រាប់ I/O operations ដែលពិបាកជាង C# ដែលប្រើ Console.ReadLine( ) យ៉ាងងាយស្រួល។

រូបភាពទី៦បង្ហាញពី Executable របស់Java

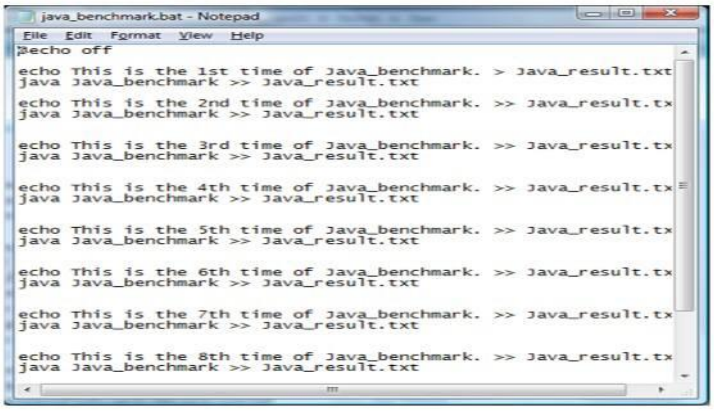


Figure 15. Screenshot of executable bat file for Java language.

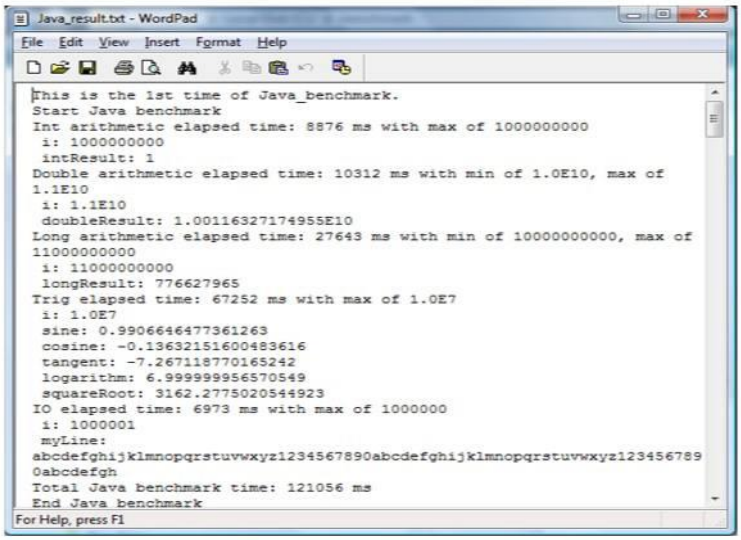


Figure 16. Screenshot of output of the executable bat files for Java language.

រូបភាពទី៧បង្ហាញពី Executableរបស់C#

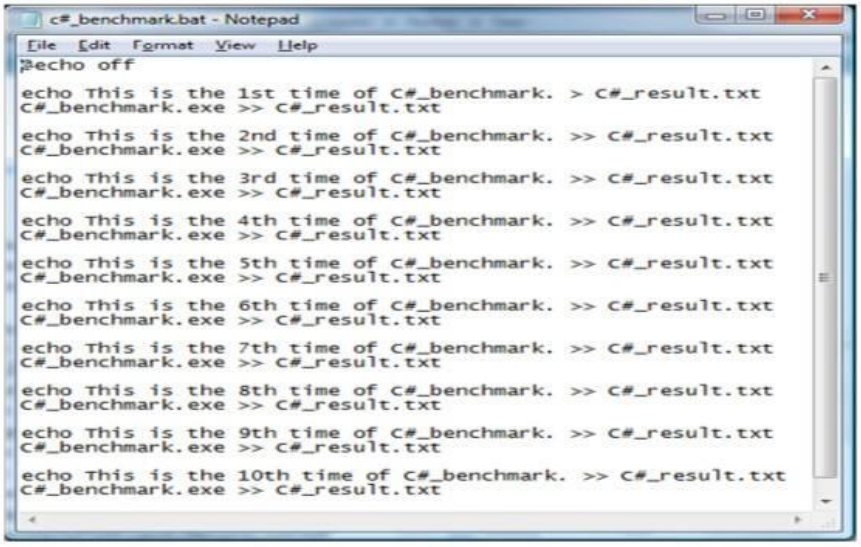


Figure 13. Screenshot of executable bat file for C# language.

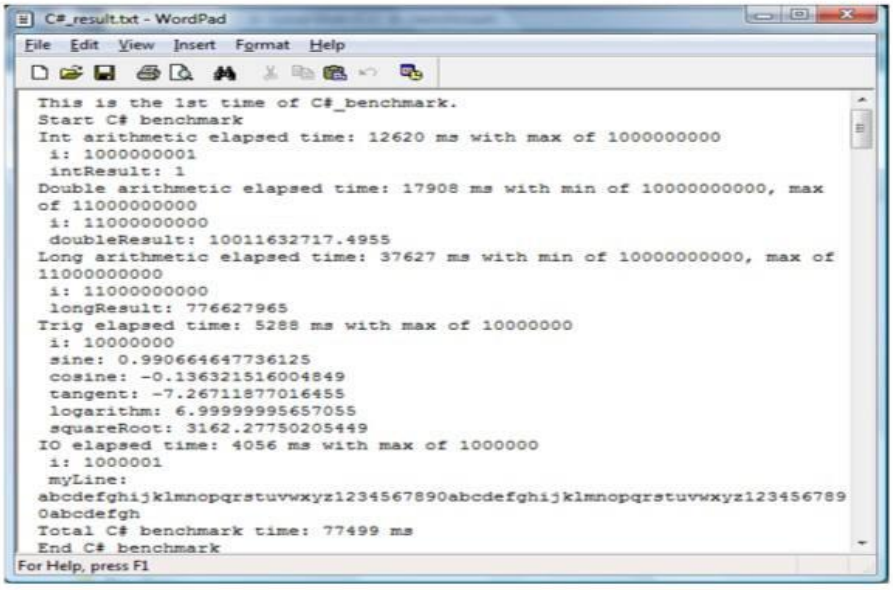


Figure 14. Screenshot of output of the executable bat files for C# language.

៣.៤ Result

រូបភាពទី៨បង្ហាញពី Result របស់ Java

Table 17. Java performance in ms.

	Int arithmetic	Double arithmetic	Long arithmetic	Trig	I/O	Total elapsed time
1st Time	8876	10312	27643	67252	6973	121056
2nd Time	8939	10343	27752	67205	6038	120277
3rd Time	8892	10281	27690	67220	6162	120245
4th Time	8908	10296	27628	67173	6240	120245
5th Time	8907	10312	27643	67158	6240	120260
6th Time	8908	10311	27659	67455	5850	120183
7th Time	8970	10374	27924	67533	5444	120245
8th Time	8907	10312	27705	67704	5476	120104
9th Time	8939	10343	27752	67798	5507	120339
10th Time	8923	10343	27768	67517	7051	121602
Average	8916.9	10322.7	27716.4	67401.5	6098.1	120455.6

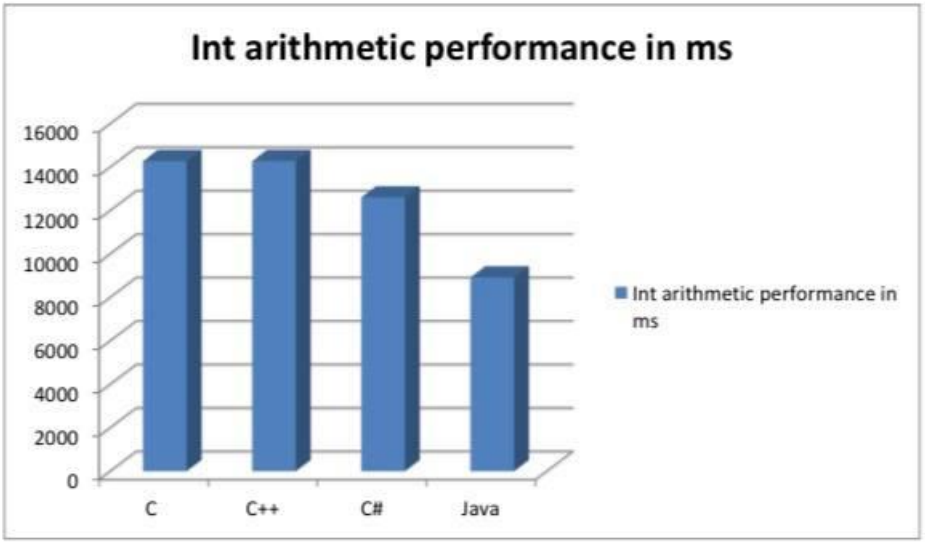
រូបភាពទី៩បង្ហាញពី Result របស់ C#

Table 16. C# performance in ms.

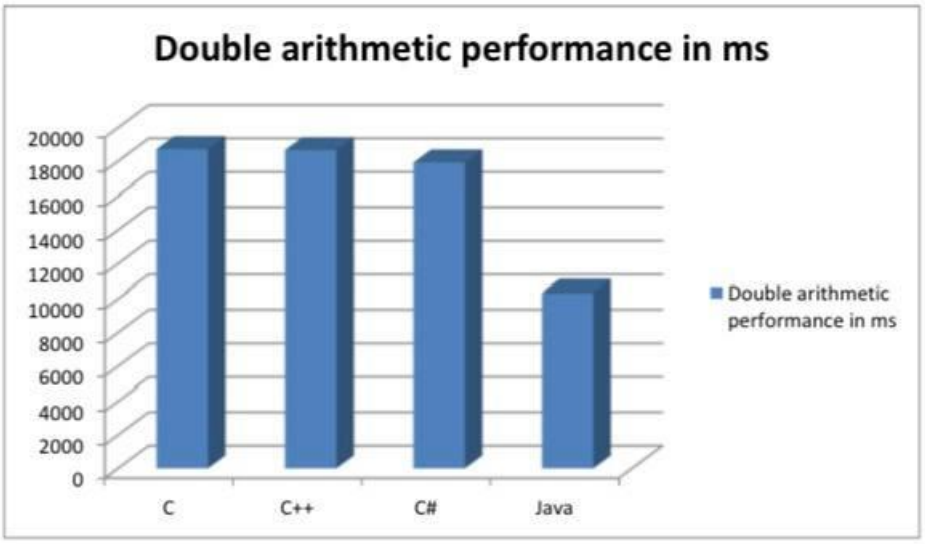
	Int arithmetic	Double arithmetic	Long arithmetic	Trig	I/O	Total elapsed time
1st Time	12620	17908	37627	5288	4056	77499
2nd Time	12589	17940	38313	5304	5382	79528
3rd Time	12636	17971	37658	5319	6708	80292
4th Time	12620	17955	37611	5397	3978	77561
5th Time	12573	17908	38188	5288	3400	77357
6th Time	12589	17924	38188	5288	3572	77561
7th Time	12589	17908	38188	5304	3369	77358
8th Time	12589	17908	38204	5304	4789	78794
9th Time	12604	17893	37596	5304	3946	77343
10th Time	12604	17893	38173	5288	3400	77358
Average	12601.3	17920.8	37974.6	5308.4	4260	78065.1

៣.៥ Analysis

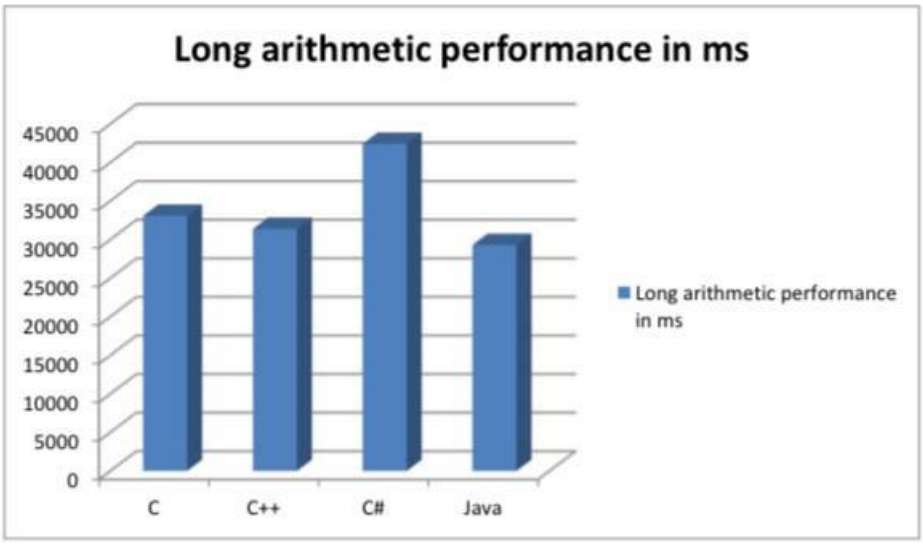
ដ្យាក្រាមទី១ បង្ហាញពី Int arithmetic performances in ms.



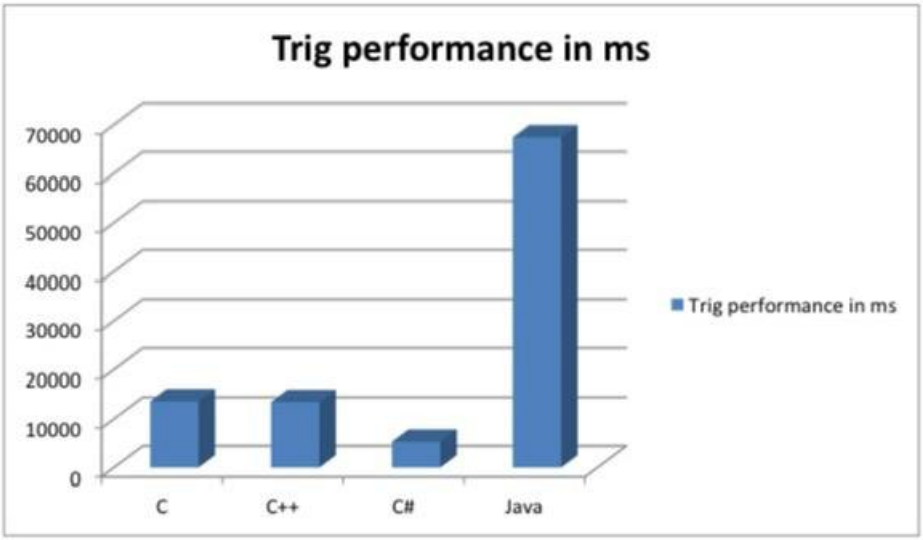
ដ្យាក្រាមទី២. Double arithmetic performances in ms.



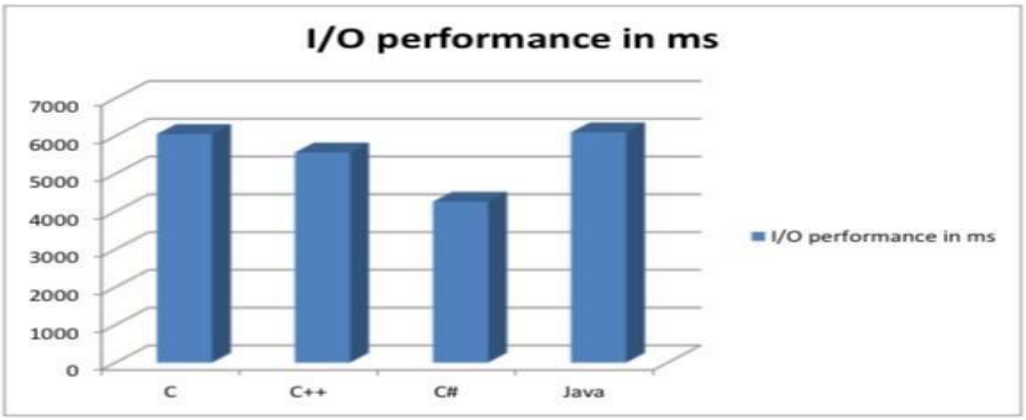
ដ្យាក្រាមទី៣ Long arithmetic performances in ms.



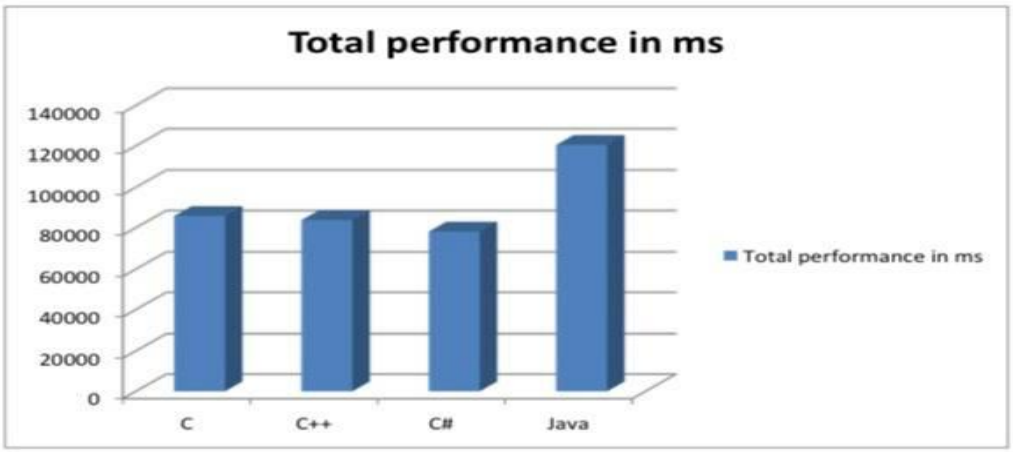
ដ្យាក្រាមទី៤. Trig arithmetic performances in ms.



ដ្យាក្រាមទី៥ . I/O performances testing in ms.



ដ្យាក្រាមទី៦. Total performances in ms.



- ភាពខុសគ្នារវាង Java និង C#
  - C# គឺល្អបំផុតក្នុងការធ្វើតេស្តនព្វន្ធ Int នព្វន្ធទ្វេ នព្វន្ធត្រីកោណ និង I/O និងដំណើរការសរុប។ ជាពិសេសនៅក្នុងការធ្វើតេស្តនព្វន្ធបី និង I/O ការអនុវត្តរបស់ C# គឺប្រសើរជាងភាសាសរសេរកម្មវិធីបីផ្សេងទៀត។ ទោះយ៉ាងណាក៏ដោយ វាមានដំណើរការអាក្រក់បំផុតនៅក្នុងនព្វន្ធឡុង។
  - លទ្ធផលនៃការអនុវត្តគោលនៃ Java គឺគួរឱ្យចាប់អារម្មណ៍ណាស់។ Java មានដំណើរការល្អបំផុតនៅក្នុងការធ្វើតេស្តនព្វន្ធ Int ការធ្វើតេស្តនព្វន្ធទ្វេ និងការធ្វើតេស្តនព្វន្ធវែង ដែលជាបីផ្នែកក្នុងចំណោមប្រាំមួយផ្នែកនៃគោល។ ទោះជាយ៉ាងណាក៏ដោយ វាមានដំណើរការអាក្រក់បំផុត

នៅក្នុងផ្នែកពីរផ្សេងទៀតនៃគោល ការធ្វើតេស្តនព្វន្ធ Trig និងការធ្វើតេស្ត I/O ។ ចាវ៉ាក៏បានដាក់ចំណាត់ថ្នាក់ចុងក្រោយដែលវាជាការអនុវត្តសរុប ដែលមានន័យថា Java ចំណាយពេលវេលាច្រើនដើម្បីអនុវត្តគោលការណ៍ជាងភាសាសរសេរកម្មវិធីបីផ្សេងទៀត។

សេចក្តីសន្និដ្ឋាន និងការផ្តល់អនុសាសន៍

១. សេចក្តីសន្និដ្ឋាន

បន្ទាប់ពីបានធ្វើការសិក្សា ស្រាវជ្រាវតាមរយៈទិន្នន័យមកបន្ទាប់ពីបានសិក្សាប្រៀបធៀបមកធ្វើអោយយើងយល់កាន់តែច្បាស់ថាភាសាJava ជាភាសាដែលពេញនិយមនិងមានប្រសិទ្ធភាពខ្ពស់សម្រាប់ការអភិវឌ្ឍកម្មវិធីតាមបណ្តាញ អ៊ីនធឺណិត និងកម្មវិធីប្រព័ន្ធផ្សេងៗ។

C# ត្រូវបានរចនាឡើងសម្រាប់សរសេរកម្មវិធី Microsoft .NET Framework ។ C# គឺជាការរួមបញ្ចូលគ្នានៃភាសាសរសេរកម្មវិធីផ្សេងទៀតទាំងអស់នៅក្នុងតុល្យភាពស្ទើរតែល្អឥតខ្ចោះ។ ការគ្រប់គ្រងអង្គចងចាំទ្រនិចក្នុង C# មិនមែនជាបញ្ហាទៀតទេ ព្រោះអ្នកប្រមូលសំរាមយកចិត្តទុកដាក់ដូចជា Java ដែរ។ ប្រព័ន្ធគ្រប់គ្រងមូលដ្ឋានទិន្នន័យទំនាក់ទំនង (RDBMS) ដូចជា Mysql, Oracle, និង Microsoft SQL Server អាចធ្វើការជាមួយ C# ដោយដំណើរការតភ្ជាប់សាមញ្ញ។ម៉្យាងវិញទៀត C# មានគុណវិបត្តិ។ C# មិនអាចបត់បែនបានទេ។ C# ពឹងផ្អែកយ៉ាងខ្លាំងលើក្របខ័ណ្ឌ .NET ។ បើគ្មាន.NET C# គឺពិបាកក្នុងការអនុវត្ត។C# អាចត្រូវបានអនុវត្តទៅនឹងការអភិវឌ្ឍកម្មវិធីដោយសារតែ C# ជាភាសាអភិវឌ្ឍន៍កម្មវិធីរហ័ស (RAD) ។ វាអាចកាត់បន្ថយរយៈពេលនៃការអភិវឌ្ឍន៍កម្មវិធីយ៉ាងខ្លាំង។ លើសពីនេះ C# ក៏សំក្តិសមបំផុតសម្រាប់ការអភិវឌ្ឍន៍កម្មវិធីគេហទំព័រ ពីព្រោះ C# មានក្របខ័ណ្ឌធំនៃសមាសធាតុដែលបានបង្កើតជាមុន ដែលអាចសម្រួលកូដនៃកម្មវិធីគេហទំព័រ។

Java គឺជាភាសាសរសេរកម្មវិធីតម្រង់ទិសវត្ថុសុទ្ធ។ វាធ្វើឱ្យកម្មវិធីម៉ូឌុលមានដើម្បីប្រើកូដឡើងវិញ។ Java គឺជាប្រភពបើកចំហ។ មនុស្សអាចប្រើវាដោយឥតគិតថ្លៃ។ វាក៏ជាវេទិកាឯករាជ្យផងដែរ ដែលជាគុណសម្បត្តិដ៏សំខាន់បំផុតមួយនៃ Java ។ កម្មវិធីដែលសរសេរក្នុង Java អាចផ្លាស់ទីបានយ៉ាងងាយស្រួលពីប្រព័ន្ធកុំព្យូទ័រមួយទៅប្រព័ន្ធមួយទៀត។Java ក៏មានគុណវិបត្តិមួយចំនួនផងដែរ។ Java គឺជាភាសាសរសេរកម្មវិធីដែលប្រើប្រាស់អង្គចងចាំ។ Java គឺយឺតព្រោះវាមានស្រទាប់បន្ថែមរវាងប្រព័ន្ធ និងកម្មវិធី។ ស្រទាប់បន្ថែមគឺ Java Virtual Machine (JVM)។ អ្វីក៏ដោយដែលធ្វើដោយកម្មវិធី Java ត្រូវតែប្រតិបត្តិដោយ Java Virtual Machine ។ បន្ទាប់មកវាធ្វើឱ្យប្រព័ន្ធធ្វើការណែនាំជាក់ស្តែង។Java មានទម្រង់បីផ្សេងគ្នាគឺ Java 2 Standard Edition (J2SE), Java2 Micro Edition (J2ME) និង Java2 Enterprise Edition (J2EE) ដែលមានលក្ខណៈស្រដៀងនឹងអ្វីដែលយើងមាននៅក្នុងប្រព័ន្ធប្រតិបត្តិការ Windows ដូចជា Windows Vista Home Basic Edition, Windows Vista Business Edition និង Windows Vista Ultimate Edition ។ ទម្រង់នីមួយៗនៃ Java មានវាលកម្មវិធីសមរម្យរបស់វា។ J2SE ដែលត្រូវបានគេហៅថា CORE Java គឺសមរម្យសម្រាប់កម្មវិធីកុំព្យូ

ទ័រ។ J2ME ត្រូវបានប្រើជាចម្បងនៅក្នុងការអភិវឌ្ឍន៍ប្រព័ន្ធដែលបានបង្កប់ ដូចជាទូរស័ព្ទចល័ត កម្មវិធីឥតខ្សែ និងកម្មវិធី PDA ។ J2EE ដែលត្រូវបានរចនាឡើងសម្រាប់កម្មវិធីសហគ្រាស ត្រូវបានប្រើជាចម្បងសម្រាប់ការអភិវឌ្ឍន៍កម្មវិធីបណ្តាញចែកចាយ ដូចជាគេហទំព័រពាណិជ្ជកម្មអេឡិចត្រូនិក និងប្រព័ន្ធ ERP ។

សរុបមក គេអាចសន្និដ្ឋានបានថា ភាសាសរសេរកម្មវិធី C# និង Java មានគុណសម្បត្តិ និងគុណវិបត្តិ។ វាពិតជាពិបាកនិយាយថាមួយណាល្អជាងអ្នកដទៃ ហើយមួយណាលឿនជាងអ្នកដទៃ។ ប៉ុន្តែភាសាសរសេរកម្មវិធីទាំងនេះមានវាលដែលសមរម្យបំផុតសម្រាប់អនុវត្ត។ មនុស្សអាចទទួលបានដំណើរការលឿន និងមានស្ថេរភាពពីកម្មវិធីដែលសរសេរជាភាសាសរសេរកម្មវិធីសមរម្យ។

២. ការផ្តល់អនុសាសន៍

ដើម្បីធ្វើការសិក្សាប្រៀបធៀបនៅភាសាសរសេរកម្មវិធី C# និង Java បានល្អប្រសើ យើងខ្ញុំសូមចែករំលែក និងផ្តល់នៅអនុសាសន៍មួយចំនួនដូចខាងក្រោម៖

- ដំបូងត្រូវធ្វើការសិក្សាស្វែងយល់ពីភាសារពីភាសា C# និង Java អោយបានល្អប្រសើរ យល់ដឹងពី Syntax , Structure របស់ភាសាទាំងពីរ និង ដឹងពីលក្ខណៈពិសេសរបស់ភាសារទាំងពីរ ។
- ធ្វើការសាកល្បង Data Types និង Primitive Operations សរសេរកូដក្នុង C# និង Java ដើម្បីធ្វើការសិក្សាលើប្រព័ន្ធ Data Types ដែលទាំងពីរនេះមានដូចជា int, long, និងប្រតិបត្តិការមូលដ្ឋាន។
- ការប្រើប្រាស់ I/O Operations ធ្វើការសាកល្បងនូវ Input/Output Operations ដើម្បីសិក្សាប្រៀបធៀបអំពីល្បឿននិងភាពងាយស្រួលក្នុងការអនុវត្តកូដក្នុងភាសាទាំងពីរ។ ត្រូវសរសេរកូដដែលអាចបង្ហាញពីលទ្ធភាពនៃដំណើរការនេះ។
- ការអនុវត្តនិងតេស្តកម្មវិធី ដោយប្រើកម្មវិធីដែលទាក់ទងនឹងប្រតិបត្តិការ (Operations) ដូចជា I/O។ អាចប្រើ Tools ដូចជា Visual Studio សម្រាប់ C# និង Eclipse ឬ IntelliJ សម្រាប់ Java ដើម្បីស្វែងយល់ពីពេលវេលាដំណើរការ និងភាពប្រសើរនៃកម្មវិធី។
- វិភាគលើគុណសម្បត្តិនិងគុណវិបត្តិរបស់ភាសាទាំងពីរ ហើយបញ្ចប់ជាមួយនឹងសន្និដ្ឋានអំពីភាពល្អ និងភាពខ្សោយនៃការប្រើប្រាស់ក្នុងស្ថានភាពផ្តាល់ខ្លួន។

ឯកសារយោង

https://www.geeksforgeeks.org/what-is-machine-language/

https://konkhmerit7.blogspot.com/2014/12/java.html

https://www.theseus.fi/handle/10024/16995

https://www.scribd.com/document/471868744/%E1%9E%98%E1%9F%81%E1%9E%9A%E1%9F%80%E1%9E%93-Java-Programming-%E1%9E%97%E1%9E%B6%E1%9E%9F%E1%9E%B6%E1%9E%81-%E1%9E%98%E1%9F%82%E1%9E%9A-pdf

https://www.bairesdev.com/blog/c-sharp-vs-java/#

https://en.wikipedia.org/wiki/C_Sharp_(programming_language)

https://www.programiz.com/csharp-programming/keywords-identifiers#:~:text=C%23%20has%20a%20total%20of,list%20of%20all%20C%23%20keywords.&text=Although%20keywords%20are%20reserved%20words,if%20%40%20is%20added%20as%20prefix

https://media.licdn.com/dms/image/C4E12AQFHYeU8N2ZDCw/article-cover_image-shrink_600_2000/0/1651746917137?e=2147483647&v=beta&t=2k4vyjwZDgK2VhoPnocHqCnAqw4vDzh12E8DJ0IEfOo

https://images.app.goo.gl/QxdqfCqbPwTQrkxu7

https://www.programiz.com/csharp-programming/keywords-identifiers#:~:text=C%23%20has%20a%20total%20of,list%20of%20all%20C%23%20keywords.&text=Although%20keywords%20are%20reserved%20words,if%20%40%20is%20added%20as%20prefix

ឧបសម្ព័ន្ធ

- Result of running C# benchmark

```
Start C# benchmark
Int arithmetic elapsed time: 12620 ms with max of 1000000000
 i: 1000000001
 intResult: 1
Double arithmetic elapsed time: 17908 ms with min of 10000000000, max of
11000000000
 i: 11000000000
 doubleResult: 10011632717.4955
Long arithmetic elapsed time: 37627 ms with min of 10000000000, max of
11000000000
 i: 11000000000
 longResult: 776627965
Trig elapsed time: 5288 ms with max of 10000000
 i: 10000000
 sine: 0.990664647736125
 cosine: -0.136321516004849
 tangent: -7.26711877016455
 logarithm: 6.99999995657055
 squareRoot: 3162.27750205449
IO elapsed time: 4056 ms with max of 1000000
 i: 1000001
 myLine:
abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwxyz1234567
890abcdefgh
Total C# benchmark time: 77499 ms
End C# benchmark
This is the 2nd time of C#_benchmark.
Start C# benchmark
Int arithmetic elapsed time: 12589 ms with max of 1000000000
 i: 1000000001
 intResult: 1
Double arithmetic elapsed time: 17940 ms with min of 10000000000, max of
11000000000
 i: 11000000000
 doubleResult: 10011632717.4955
Long arithmetic elapsed time: 38313 ms with min of 10000000000, max of
11000000000
 i: 11000000000
 longResult: 776627965
Trig elapsed time: 5304 ms with max of 10000000
 i: 10000000
```

```
 sine: 0.990664647736125
 cosine: -0.136321516004849
 tangent: -7.26711877016455
 logarithm: 6.99999995657055
 squareRoot: 3162.27750205449
IO elapsed time: 5382 ms with max of 1000000
 i: 1000001
 myLine:
abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwxyz1234567
890abcdefgh
Total C# benchmark time: 79528 ms
End C# benchmark
This is the 3rd time of C#_benchmark.
Start C# benchmark
Int arithmetic elapsed time: 12636 ms with max of 1000000000
 i: 1000000001
 intResult: 1
Double arithmetic elapsed time: 17971 ms with min of 10000000000, max of
11000000000
 i: 11000000000
 doubleResult: 10011632717.4955
Long arithmetic elapsed time: 37658 ms with min of 10000000000, max of
11000000000
 i: 11000000000
 longResult: 776627965
Trig elapsed time: 5319 ms with max of 10000000
 i: 10000000
 sine: 0.990664647736125
 cosine: -0.136321516004849
 tangent: -7.26711877016455
 logarithm: 6.99999995657055
 squareRoot: 3162.27750205449
IO elapsed time: 6708 ms with max of 1000000
 i: 1000001
 myLine:
abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwxyz1234567
890abcdefgh
Total C# benchmark time: 80292 ms
End C# benchmark
This is the 4th time of C#_benchmark.
Start C# benchmark
Int arithmetic elapsed time: 12620 ms with max of 1000000000
 i: 1000000001
 intResult: 1
```

```
Double arithmetic elapsed time: 17955 ms with min of 10000000000, max of
11000000000
 i: 11000000000
 doubleResult: 10011632717.4955
Long arithmetic elapsed time: 37611 ms with min of 10000000000, max of
11000000000
 i: 11000000000
 longResult: 776627965
Trig elapsed time: 5397 ms with max of 10000000
 i: 10000000
 sine: 0.990664647736125
 cosine: -0.136321516004849
 tangent: -7.26711877016455
 logarithm: 6.99999995657055
 squareRoot: 3162.27750205449
IO elapsed time: 3978 ms with max of 1000000
 i: 1000001
 myLine:
abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwxyz1234567
890abcdefgh
Total C# benchmark time: 77561 ms
End C# benchmark
This is the 5th time of C#_benchmark.
Start C# benchmark
Int arithmetic elapsed time: 12573 ms with max of 1000000000
 i: 1000000001
 intResult: 1
Double arithmetic elapsed time: 17908 ms with min of 10000000000, max of
11000000000
 i: 11000000000
 doubleResult: 10011632717.4955
Long arithmetic elapsed time: 38188 ms with min of 10000000000, max of
11000000000
 i: 11000000000
 longResult: 776627965
Trig elapsed time: 5288 ms with max of 10000000
 i: 10000000
 sine: 0.990664647736125
 cosine: -0.136321516004849
 tangent: -7.26711877016455
 logarithm: 6.99999995657055
 squareRoot: 3162.27750205449
IO elapsed time: 3400 ms with max of 1000000
 i: 1000001
```

```
 myLine:
abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwxyz1234567
890abcdefgh
Total C# benchmark time: 77357 ms
End C# benchmark
This is the 6th time of C#_benchmark.
Start C# benchmark
Int arithmetic elapsed time: 12589 ms with max of 1000000000
 i: 1000000001
 intResult: 1
Double arithmetic elapsed time: 17924 ms with min of 10000000000, max of
11000000000
 i: 11000000000
 doubleResult: 10011632717.4955
Long arithmetic elapsed time: 38188 ms with min of 10000000000, max of
11000000000
 i: 11000000000
 longResult: 776627965
Trig elapsed time: 5288 ms with max of 10000000
 i: 10000000
 sine: 0.990664647736125
 cosine: -0.136321516004849
 tangent: -7.26711877016455
 logarithm: 6.99999995657055
 squareRoot: 3162.27750205449
IO elapsed time: 3572 ms with max of 1000000
 i: 1000001
 myLine:
abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwxyz1234567
890abcdefgh
Total C# benchmark time: 77561 ms
End C# benchmark
This is the 7th time of C#_benchmark.
Start C# benchmark
Int arithmetic elapsed time: 12589 ms with max of 1000000000
 i: 1000000001
 intResult: 1
Double arithmetic elapsed time: 17908 ms with min of 10000000000, max of
11000000000
 i: 11000000000
 doubleResult: 10011632717.4955
Long arithmetic elapsed time: 38188 ms with min of 10000000000, max of
11000000000
 i: 11000000000
```

```
 longResult: 776627965
Trig elapsed time: 5304 ms with max of 10000000
 i: 10000000
 sine: 0.990664647736125
 cosine: -0.136321516004849
 tangent: -7.26711877016455
 logarithm: 6.99999995657055
 squareRoot: 3162.27750205449
IO elapsed time: 3369 ms with max of 1000000
 i: 1000001
 myLine:
abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwxyz1234567
890abcdefgh
Total C# benchmark time: 77358 ms
End C# benchmark
This is the 8th time of C#_benchmark.
Start C# benchmark
Int arithmetic elapsed time: 12589 ms with max of 1000000000
 i: 1000000001
 intResult: 1
Double arithmetic elapsed time: 17908 ms with min of 10000000000, max of
11000000000
 i: 11000000000
 doubleResult: 10011632717.4955
Long arithmetic elapsed time: 38204 ms with min of 10000000000, max of
11000000000
 i: 11000000000
 longResult: 776627965
Trig elapsed time: 5304 ms with max of 10000000
 i: 10000000
 sine: 0.990664647736125
 cosine: -0.136321516004849
 tangent: -7.26711877016455
 logarithm: 6.99999995657055
 squareRoot: 3162.27750205449
IO elapsed time: 4789 ms with max of 1000000
 i: 1000001
 myLine:
abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwxyz1234567
890abcdefgh
Total C# benchmark time: 78794 ms
End C# benchmark
This is the 9th time of C#_benchmark.
Start C# benchmark
```

```
Int arithmetic elapsed time: 12604 ms with max of 1000000000
 i: 1000000001
 intResult: 1
Double arithmetic elapsed time: 17893 ms with min of 10000000000, max of
11000000000
 i: 11000000000
 doubleResult: 10011632717.4955
Long arithmetic elapsed time: 37596 ms with min of 10000000000, max of
11000000000
 i: 11000000000
 longResult: 776627965
Trig elapsed time: 5304 ms with max of 10000000
 i: 10000000
 sine: 0.990664647736125
 cosine: -0.136321516004849
 tangent: -7.26711877016455
 logarithm: 6.99999995657055
 squareRoot: 3162.27750205449
IO elapsed time: 3946 ms with max of 1000000
 i: 1000001
 myLine:
abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwxyz1234567
890abcdefgh
Total C# benchmark time: 77343 ms
End C# benchmark
This is the 10th time of C#_benchmark.
Start C# benchmark
Int arithmetic elapsed time: 12604 ms with max of 1000000000
 i: 1000000001
 intResult: 1
Double arithmetic elapsed time: 17893 ms with min of 10000000000, max of
11000000000
 i: 11000000000
 doubleResult: 10011632717.4955
Long arithmetic elapsed time: 38173 ms with min of 10000000000, max of
11000000000
 i: 11000000000
 longResult: 776627965
Trig elapsed time: 5288 ms with max of 10000000
 i: 10000000
 sine: 0.990664647736125
 cosine: -0.136321516004849
 tangent: -7.26711877016455
 logarithm: 6.99999995657055
```

```
 squareRoot: 3162.27750205449
IO elapsed time: 3400 ms with max of 1000000
 i: 1000001
 myLine:
abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwxyz1234567
890abcdefgh
Total C# benchmark time: 77358 ms
End C# benchmark
```

- Result of running Java benchmark

```
Start Java benchmark
Int arithmetic elapsed time: 8876 ms with max of 1000000000
 i: 1000000000
 intResult: 1
Double arithmetic elapsed time: 10312 ms with min of 1.0E10, max of 1.1E10
 i: 1.1E10
 doubleResult: 1.00116327174955E10
Long arithmetic elapsed time: 27643 ms with min of 10000000000, max of
11000000000
 i: 11000000000
 longResult: 776627965
Trig elapsed time: 67252 ms with max of 1.0E7
 i: 1.0E7
 sine: 0.9906646477361263
 cosine: -0.13632151600483616
 tangent: -7.267118770165242
 logarithm: 6.999999956570549
 squareRoot: 3162.2775020544923
IO elapsed time: 6973 ms with max of 1000000
 i: 1000001
 myLine:
abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwxyz1234567
890abcdefgh
Total Java benchmark time: 121056 ms
End Java benchmark
This is the 2nd time of Java_benchmark.
Start Java benchmark
Int arithmetic elapsed time: 8939 ms with max of 1000000000
 i: 1000000000
 intResult: 1
Double arithmetic elapsed time: 10343 ms with min of 1.0E10, max of 1.1E10
 i: 1.1E10
 doubleResult: 1.00116327174955E10
Long arithmetic elapsed time: 27752 ms with min of 10000000000, max of
11000000000
 i: 11000000000
 longResult: 776627965
Trig elapsed time: 67205 ms with max of 1.0E7
 i: 1.0E7
 sine: 0.9906646477361263
 cosine: -0.13632151600483616
```

```
 tangent: -7.267118770165242
 logarithm: 6.999999956570549
 squareRoot: 3162.2775020544923
IO elapsed time: 6038 ms with max of 1000000
 i: 1000001
 myLine:
abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwxyz1234567
890abcdefgh
Total Java benchmark time: 120277 ms
End Java benchmark
This is the 3rd time of Java_benchmark.
Start Java benchmark
Int arithmetic elapsed time: 8892 ms with max of 1000000000
 i: 1000000000
 intResult: 1
Double arithmetic elapsed time: 10281 ms with min of 1.0E10, max of 1.1E10
 i: 1.1E10
 doubleResult: 1.00116327174955E10
Long arithmetic elapsed time: 27690 ms with min of 10000000000, max of
11000000000
 i: 11000000000
 longResult: 776627965
Trig elapsed time: 67220 ms with max of 1.0E7
 i: 1.0E7
 sine: 0.9906646477361263
 cosine: -0.13632151600483616
 tangent: -7.267118770165242
 logarithm: 6.999999956570549
 squareRoot: 3162.2775020544923
IO elapsed time: 6162 ms with max of 1000000
 i: 1000001
 myLine:
abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwxyz1234567
890abcdefgh
Total Java benchmark time: 120245 ms
End Java benchmark
This is the 4th time of Java_benchmark.
Start Java benchmark
Int arithmetic elapsed time: 8908 ms with max of 1000000000
 i: 1000000000
 intResult: 1
Double arithmetic elapsed time: 10296 ms with min of 1.0E10, max of 1.1E10
 i: 1.1E10
 doubleResult: 1.00116327174955E10
```

```
This is the 6th time of Java_benchmark.
Start Java benchmark
Int arithmetic elapsed time: 8908 ms with max of 1000000000
 i: 1000000000
 intResult: 1
Double arithmetic elapsed time: 10311 ms with min of 1.0E10, max of 1.1E10
 i: 1.1E10
 doubleResult: 1.00116327174955E10
Long arithmetic elapsed time: 27659 ms with min of 10000000000, max of
11000000000
 i: 11000000000
 longResult: 776627965
Trig elapsed time: 67455 ms with max of 1.0E7
 i: 1.0E7
 sine: 0.9906646477361263
 cosine: -0.13632151600483616
 tangent: -7.267118770165242
 logarithm: 6.999999956570549
 squareRoot: 3162.2775020544923
IO elapsed time: 5850 ms with max of 1000000
 i: 1000001
 myLine:
abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwxyz1234567
890abcdefgh
Total Java benchmark time: 120183 ms
End Java benchmark
This is the 7th time of Java_benchmark.
Start Java benchmark
Int arithmetic elapsed time: 8970 ms with max of 1000000000
 i: 1000000000
 intResult: 1
Double arithmetic elapsed time: 10374 ms with min of 1.0E10, max of 1.1E10
 i: 1.1E10
 doubleResult: 1.00116327174955E10
Long arithmetic elapsed time: 27924 ms with min of 10000000000, max of
11000000000
 i: 11000000000
 longResult: 776627965
Trig elapsed time: 67533 ms with max of 1.0E7
 i: 1.0E7
 sine: 0.9906646477361263
 cosine: -0.13632151600483616
 tangent: -7.267118770165242
 logarithm: 6.999999956570549
```

```
 i: 11000000000
 longResult: 776627965
Trig elapsed time: 67798 ms with max of 1.0E7
 i: 1.0E7
 sine: 0.9906646477361263
 cosine: -0.13632151600483616
 tangent: -7.267118770165242
 logarithm: 6.999999956570549
 squareRoot: 3162.2775020544923
IO elapsed time: 5507 ms with max of 1000000
 i: 1000001
 myLine:
abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwxyz1234567
890abcdefgh
Total Java benchmark time: 120339 ms
End Java benchmark
This is the 10th time of Java_benchmark.
Start Java benchmark
Int arithmetic elapsed time: 8923 ms with max of 1000000000
 i: 1000000000
 intResult: 1
Double arithmetic elapsed time: 10343 ms with min of 1.0E10, max of 1.1E10
 i: 1.1E10
 doubleResult: 1.00116327174955E10
Long arithmetic elapsed time: 27768 ms with min of 10000000000, max of
11000000000
 i: 11000000000
 longResult: 776627965
Trig elapsed time: 67517 ms with max of 1.0E7
 i: 1.0E7
 sine: 0.9906646477361263
 cosine: -0.13632151600483616
 tangent: -7.267118770165242
 logarithm: 6.999999956570549
 squareRoot: 3162.2775020544923
IO elapsed time: 7051 ms with max of 1000000
 i: 1000001
 myLine:
abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwxyz1234567
890abcdefgh
Total Java benchmark time: 121602 ms
End Java benchmark
```