

ជំពូកទី ៨ : ការប្រើ Exceptions

អ្វីទៅ Exception?

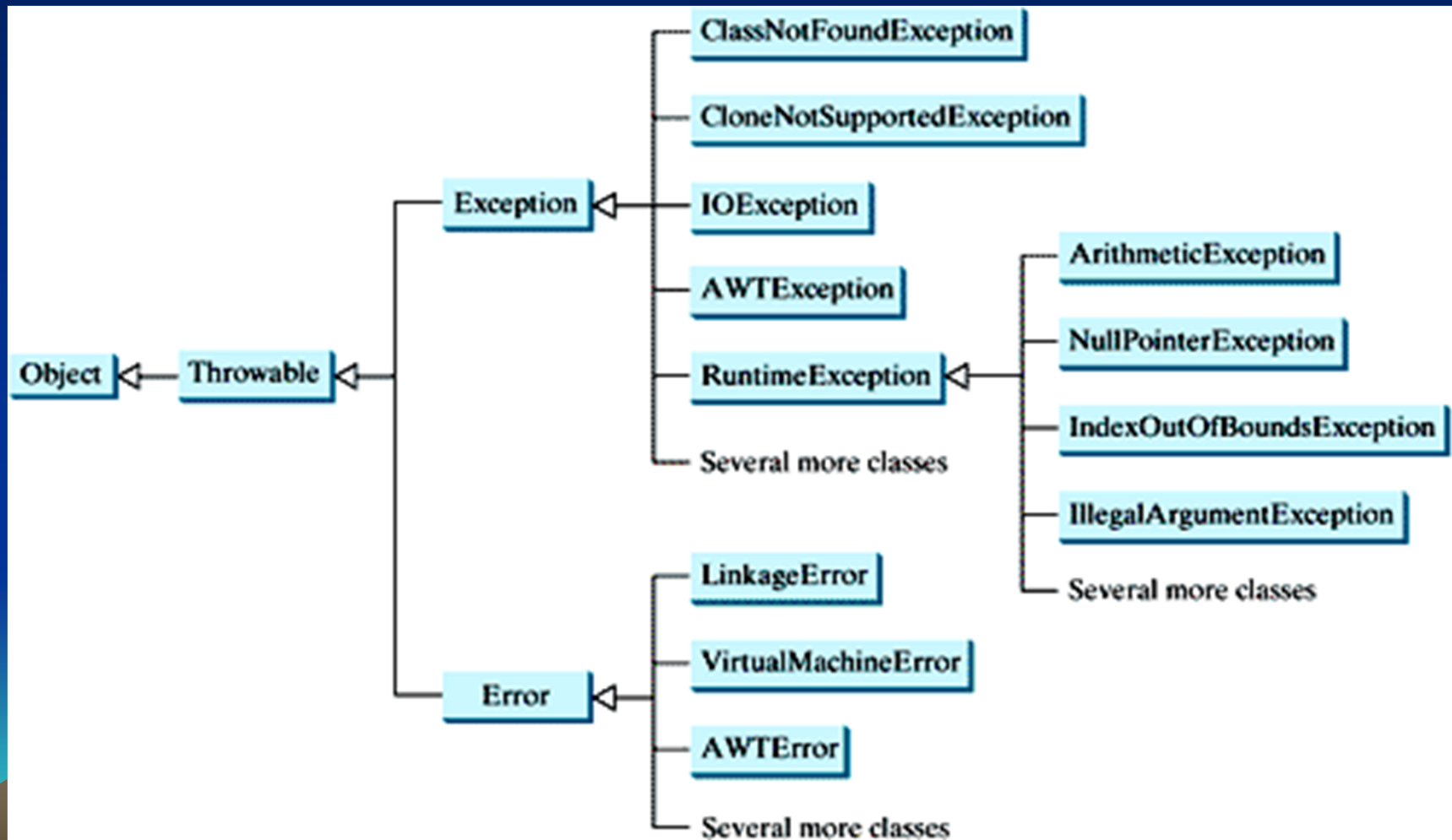
Exception មួយគឺជា error មួយប្រភេទដែលកើតឡើងនៅលើ code នៃកម្មវិធីក្នុងពេលដំណើរការក្រោមស័ក្ខខ័ណ្ឌមិនប្រក្រតី ។

ឧទាហរណ៍: ការចែកនឹងសូន្យឬការរកឯកសារមិនឃើញជាដើម ។

១. លំដាប់ថ្នាក់នៃ Exceptions

គ្រប់ exceptions ទាំងអស់តាងអោយ classes នៅក្នុង Java ។ exception classes ទាំងអស់មានប្រភព ពី Throwable class ។ Throwable មាន subclass ពីរគឺ Exception class និង Error class ។ exceptions នៃប្រភេទ Error ទាក់ទងទៅនឹង error ដែលកើតឡើងនៅក្នុង Java virtual machine ខ្លួនវា ហើយមិនមាននៅក្នុងកម្មវិធីរបស់យើងទេ ។ រីឯ Exception ជា error ដែលបានមកពីសកម្មភាពកម្មវិធី ។

ឧទាហរណ៍: ការចែកនឹងសូន្យ, ហួសដែនកំណត់របស់ array និង file errors ជាដើម ។



២. មូលដ្ឋានគ្រឹះនៃការប្រើ Exception

ការប្រើ Java exception ត្រូវបានគ្រប់គ្រងតាម
រយៈពាក្យ : try, catch, throw, throws និង finally ។
វាបង្កើតបានជាប្រព័ន្ធតូចៗយ៉ាងមាំមួន ។

- try block សំរាប់ឃ្លាំមើល code ដែលមាន errors
- catch block សំរាប់ចាប់ error ទៅតាមប្រភេទរបស់វា

- **throw** សំរាប់ធ្វើអោយកើតមាន error ប្រភេទណាមួយ
តាមដែលគេសរសេរ ដើម្បីចាប់គ្រវែងចោលនូវ error នោះ
- **throws** សំរាប់ចាប់គ្រវែងចោលនូវ error អោយចេញផុត ពី
method មួយ
- **finally block** សំរាប់ផ្តល់នូវព័ត៌មានចំពោះ code ណា
ដែលតែងតែប្រតិបត្តិការនូវអ្វីមួយពី try block នោះ ។

២.១ ការប្រើ try និង catch

ទំរង់ទូទៅនៃការប្រើ try/catch block :

```
try {  
    // block of code to monitor for errors  
}  
catch (ExceptionType1 exOb) {  
    // handler for ExceptionType1  
}  
catch (ExceptionType2 exOb) {  
    // handler for ExceptionType2  
}  
.....  
.....
```

ក្នុងនេះ *ExceptionType* គឺជាប្រភេទ
exception ដែលបានកើតឡើង ។

២.២ ដលវិធាននៃការមិនប្រើ Exceptions

បើកម្មវិធីរបស់យើងមិនធ្វើការចាប់យក exception ណាមួយ ពេលនោះវានឹងមានការចាប់ exception ដោយ JVM ។
បញ្ហាចោទនោះគឺ ការប្រើ exception ធម្មតា
បញ្ចប់ការប្រតិបត្តិ និងបង្ហាញនូវឃ្លាមានកំហុស ។

ឧទាហរណ៍ : វាមាន error អំពី index របស់ array
កើតឡើងនោះ ការប្រតិបត្តិត្រូវបញ្ឈប់រួចហើយវាបង្ហាញនូវ
ឃ្លាមានកំហុសដូចខាងក្រោមនេះ :


```
// Let JVM handle the error.  
class NotHandled {  
    public static void main(String args[]) {  
  
        int nums[] = new int[4];  
        System.out.println("Before exception  
                             is generated.");  
  
        // generate an index out-of-bounds  
        // exception  
        nums[7] = 10;  
    }  
}
```

ពេលនោះ error លេចឡើងនូវឃ្លាមានកំហុសដូចខាងក្រោម :



ដូចបានរៀបរាប់ខាងដើមមកហើយថាប្រភេទ exception ត្រូវតែស្ថិតានឹងប្រភេទ exception ដែលបានកំណត់ក្នុងឃ្លា catch ។ បើមិនដូច្នោះទេនោះ exception មិនត្រូវបានចាប់យកឡើយ ។

ឧទាហរណ៍ : កម្មវិធីខាងក្រោមនេះព្យាយាមចាប់យក error ប្រភេទ index នៃ array ជាមួយឃ្លា catch ចំពោះ ArithmeticException ។ 

នៅពេលធ្វើការ compile នោះ លទ្ធផលបង្ហាញដូចខាងក្រោមនេះ : 

៣. ការប្រើប្រាស់ catch ច្រើន

គេអាចប្រើប្រាស់ catch ច្រើនបានជាមួយនឹង try តែមួយ ។
ទោះជាយ៉ាងណាក៏ដោយ ប្រាស់ catch នីមួយៗត្រូវតែចាប់យក
នូវប្រភេទ exception ផ្សេងគ្នា ។

ឧទាហរណ៍ : កម្មវិធីខាងក្រោមនេះបង្ហាញពីការចាប់យក
errors ក្នុង array ដែលប្រើហួសការកំណត់ ហើយនិង
ការចែកចំនួនមួយទៅនឹងសូន្យ ។



៤. try block មួយនៅក្នុង try block មួយទៀត

try មួយអាចស្ថិតនៅក្នុង try មួយទៀតបាន ។ exception មួយបានបង្កើតឡើងនៅក្នុង try block ផ្នែកខាងក្នុងដែលមិនត្រូវបានចាប់យកដោយ catch ហើយមានទំនាក់ទំនងជាមួយនឹង try block នោះ ។

ឧទាហរណ៍ : នៅក្នុងកម្មវិធី 

ArrayIndexOutOfBoundsException មិនត្រូវបានចាប់យកដោយ try block នៅផ្នែកខាងក្នុងឡើយ ប៉ុន្តែវាត្រូវចាប់យកដោយ try នៅផ្នែកខាងក្រៅទៅវិញ ។

៥. ការធ្វើអោយកកើតឡើងនូវ exception

គេអាចធ្វើអោយមានការកកើតឡើងនូវ exception មួយតាមលក្ខណៈធម្មតាដោយប្រើ throw ។ ទំរង់ទូទៅរបស់វា

គឺ៖ *throw exceptOb;*

ដែលក្នុងនេះ *exceptOb* គឺជា object នៃ exception class

មួយបានទទួលលក្ខណៈពី Throwable ។

ឧទាហរណ៍ : កម្មវិធីខាងក្រោមនេះបង្ហាញពីការប្រើប្រាស់ throw ដែលធ្វើអោយកកើតឡើងនូវ ArithmeticException តាម

លក្ខណៈធម្មតា ។



៦. ការប្រើ finally

ដើម្បីកំណត់ទំហំ block នៃ code សំរាប់ប្រតិបត្តិការ
នៅពេល try/catch block បានចាកចេញផុតមិនថាទោះជា
នៅក្នុងលក្ខខណ្ឌណាក៏ដោយ ។ ទំរង់ទូទៅរបស់វា គឺ៖

```
try {  
    // block of code to monitor for errors  
} catch (ExceptionType1 exOb) {  
    // handler for ExceptionType1  
}  
// ...  
finally {  
    // finally code  
}
```



៧. ការប្រើ throws

បើ method មួយបណ្តាលអោយមាន exception មួយ ដែលវាមិនទាន់ប្រើនោះ វាត្រូវតែបញ្ជាក់នូវលក្ខណៈនេះ ហេតុនេះ អ្នកប្រើ method នេះអាចយល់មើលនូវ exceptions ដើម្បី ទប់ទល់នឹងវា ។

ទំរង់ទូទៅនៃការប្រកាស method ដែលមានឃ្លា throws :

ret-type methName(param-list) throws

except-list {

// body

}



៨. ការបង្កើត Exception subclass

Java អាចអោយគេបង្កើត exception ប្រើទៅតាម error ក្នុង code នៃកម្មវិធីរបស់ខ្លួន។ ការបង្កើត exception មួយមានលក្ខណៈងាយណាស់ គឺគ្រាន់តែកំណត់លក្ខណៈអោយ subclass របស់ Exception (ដែលជា subclass នៃ Throwable)។ Subclass មិនចាំបាច់អនុវត្ត ឬប្រើអ្វីផ្សេងទៀតឡើយ ពោលគឺវាអាចអោយគេប្រើបានដូច exceptions ធម្មតាដែរ។ ឧទាហរណ៍ :



៩. Exception មានត្រូវរាប់រមស់ Java

Java បានកំណត់នូវពពួក exception classes ជាច្រើន នៅក្នុង package ទូទៅមួយឈ្មោះ java.lang ។ exceptions ដែលប្រើជាទូទៅភាគច្រើនគឺជា subclasses នៃប្រភេទ RuntimeException ។ ពពួក exceptions ទាំងនេះត្រូវបានហៅថា *unchecked exceptions* ព្រោះ compiler មិនធ្វើការត្រួតពិនិត្យទៅលើ method ថាតើវា បានប្រើ ឬធ្វើអោយមានការកកើតឡើងនូវ exceptions ដែរ

ឬទេ ។

ក្រៅពីនេះពួក exceptions ត្រូវបានហៅថា *checked exceptions* ព្រោះ compiler នឹងធ្វើការត្រួតពិនិត្យទៅលើ method នោះដោយប្រើ throws នៅពេលប្រកាស method ។
ចូរពិនិត្យមើលតារាងនៃពួក unchecked exceptions និង checked exceptions ព្រមទាំងលំដាប់ថ្នាក់នៃ exception ។



សំណួរនិងចម្លើយ

- ១ - អ្វីទៅជា exception ?
- ២ - តើ catch ប្រើសំរាប់ធ្វើអ្វី? តើ code ដែលមាន error កើតឡើងត្រូវសរសេរនៅក្នុងផ្នែកណា ដើម្បីអោយ catch ចាប់បាន?
- ៣ - ហេតុអ្វីបានជាគេចង់បង្កើត exception ធម្មតារបស់ខ្លួន?
- ៤ - ចូរបង្ហាញពីការប្រើ throw និង throws ។

- ៥ - តើមានអ្វីកើតឡើង នៅពេលដែល exception មួយ មិនបានចាប់ error?
- ៦ - តើពពួក exception classes ជា subclass របស់ class ណា?
- ៧ - តើនៅពេលណា code នៅក្នុង block របស់ finally ត្រូវបានប្រតិបត្តិការ?
- ៨ - តើភាពខុសគ្នារវាងពពួក checked exception និង unchecked exception មានអ្វីខ្លះ?