Exceptions

Ionut
Spalatelu

Outline

- Understanding exceptions
- Syntax
- try/catch/finally
- Exception types

Understanding exceptions

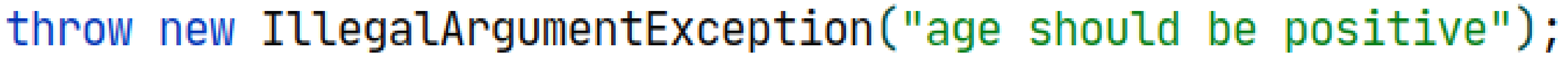
- Programs will encounter errors while they run (at runtime)
- Error = anything that prevents your program to proceed further: null reference, zero division, invalid or missing input, insufficient resources (memory, disk space)
- **Exception** = program's response when 'doesn't know' what to do, how to proceed further
- Exception object will be specific to that case that caused it

Understanding exceptions

- When an exception is thrown, a specific sequence of steps is executed by the JVM, in order to give the program a chance to recover
- If an exception is only thrown and not caught then program exits
- If an exception is thrown and caught then program may continue normal flow
- Main concepts:
  - **throwing** an exception: the way your code tells the JVM that it encountered an error
  - **catching** an exception: the way your code tells the JVM that it wants to "handle" an error.

try/catch/finally

- Throwing an exception

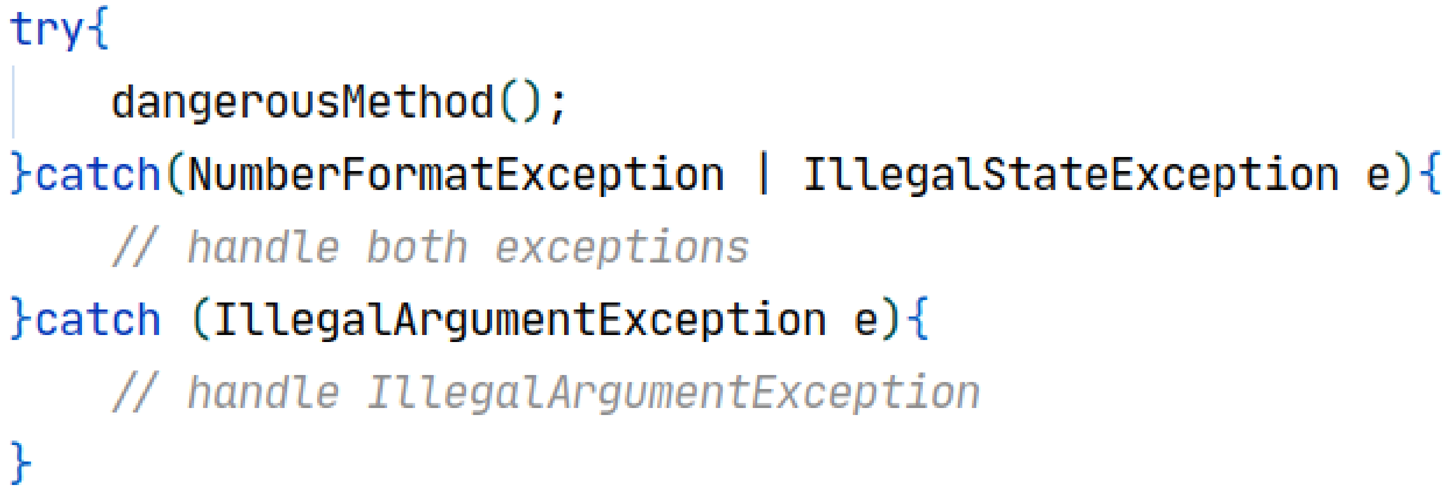


```
throw new IllegalArgumentException("age should be positive");
```

- Things to note:
  - The exception is just a normal Java object
  - When the program encounters such a statement the JVM exception handling mechanism takes control of what will be executed next

try/catch/finally

- Catching exceptions

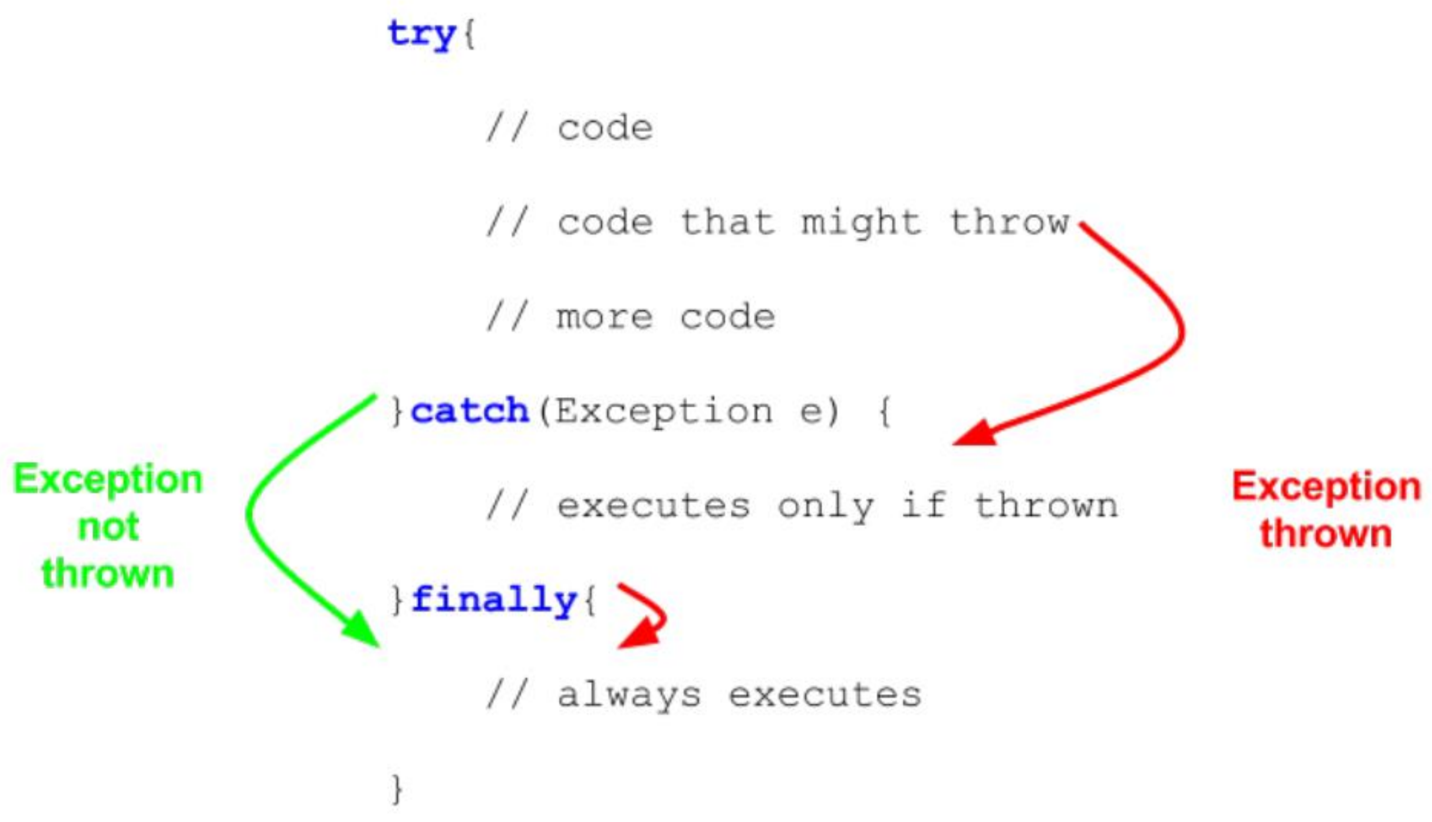


```java
try{
    dangerousMethod();
}catch(NumberFormatException | IllegalStateException e){
    // handle both exceptions
}catch (IllegalArgumentException e){
    // handle IllegalArgumentException
}
```

- Things to note:
  - The code that throws exceptions needs to be inside the try block
  - Different exception types can be handled independently
  - The order of catch clauses matters
  - A catch handles exceptions of the declared type and all its subclasses

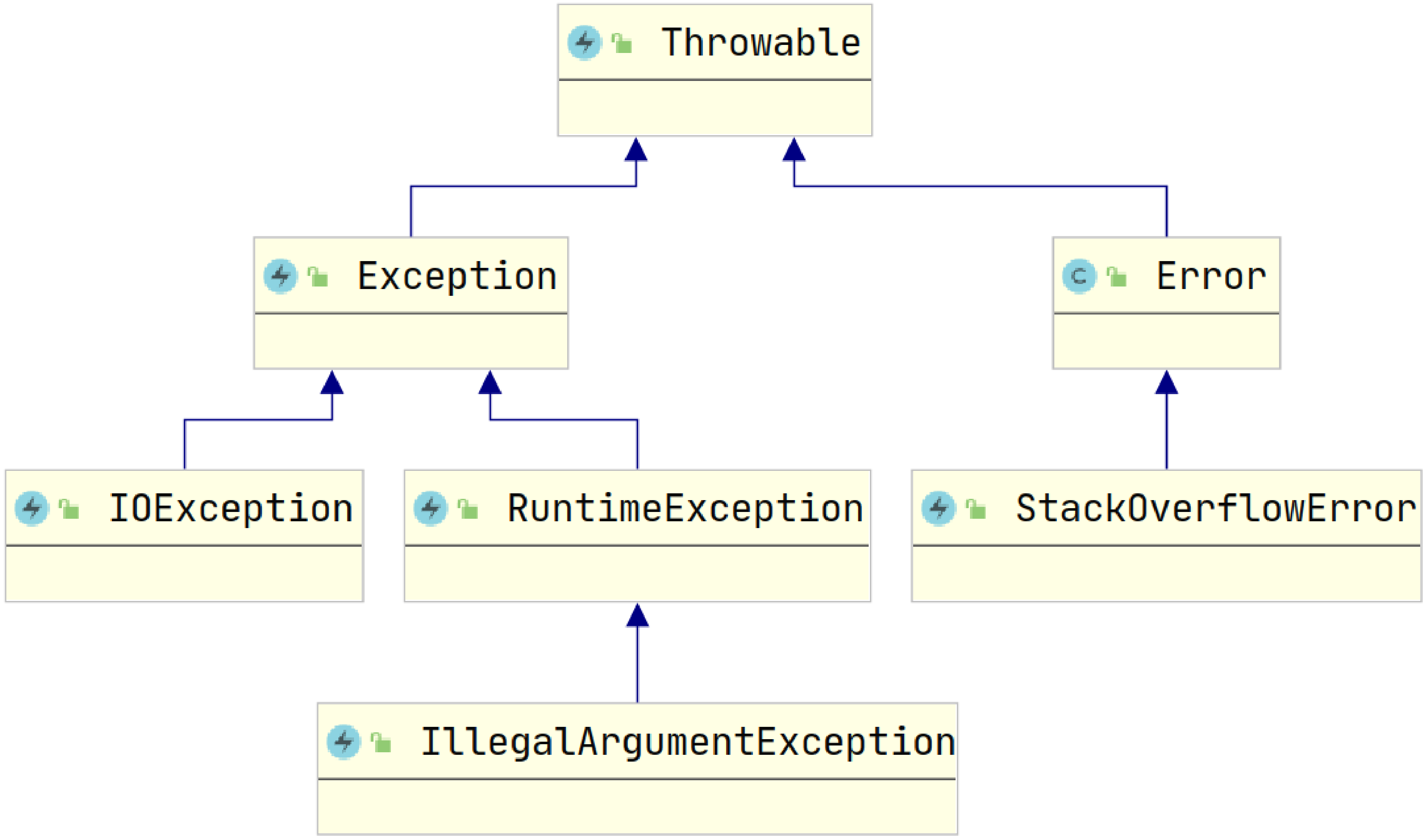
finally

- Block that always executes regardless of whether an exception has been thrown or not



```
try{
    // code
    // code that might throw
    // more code
}catch(Exception e) {
    // executes only if thrown
}finally{
    // always executes
}
```

Exception types



Exception types

In Java exceptions are splitted into 2 categories:

- checked: checked by the compiler and force to follow "handle or declare" rule. They extend directly **Exception** superclass

- unchecked: they are not checked by compiler, so the caller is not forced to take any actions. They extend **RuntimeException** superclass

Summary

What we learned:

- What exceptions are and why they are needed;
- How to throw them using **throw**
- How to handle them using **try-catch**
- How to declare them using **throws**
- Differentiate between checked and unchecked exceptions
- Create custom checked (by extending from Exception) or unchecked (by extending from RuntimeException) exceptions

Questions