COMP 110/L Lecture 22

Mahdi Ebrahimi

Slides adapted from Dr. Kyle Dewey

Outline

- Exceptions

Exceptions

Recall

```
int[] array = new int[3];
int result = array[27];
```

Recall

```
int[] array = new int[3];
int result = array[27];
```

```
Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException
```

Recall

```
int[] array = new int[3];
int result = array[27];
```

```
Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException
```

```
int result = Integer.parseInt(“hello”);
```

Recall

```
int[] array = new int[3];
int result = array[27];
```

```
Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException
```

```
int result = Integer.parseInt(“hello”);
```

```
Exception in thread "main"
java.lang.NumberFormatException
```

Exceptions

An exception is an unwanted or unexpected **event**, which occurs during the execution of a program i.e at **run time**, that disrupts the normal flow of the program.

Error: An Error indicates **serious** problem that a reasonable application should not try to catch.
Exception: Exception indicates **abnormal** conditions that a reasonable application might try to catch.

Exceptions

- Intended to signal events which happen infrequently but cannot be ignored
  - “Exceptional”
  - Errors are common examples
- Can define different kinds of exceptions for different conditions

Exceptions

- Intended to signal events which happen infrequently but cannot be ignored
  - “Exceptional”
  - Errors are common examples
- Can define different kinds of exceptions for different conditions

```
java.lang.ArrayIndexOutOfBoundsException
     java.lang.NumberFormatException
```

Exceptions Handling

Is a mechanism to handle run-time errors such as `ClassNotFoundException`, `IOException`, `SQLException`, `RemoteException`, etc.

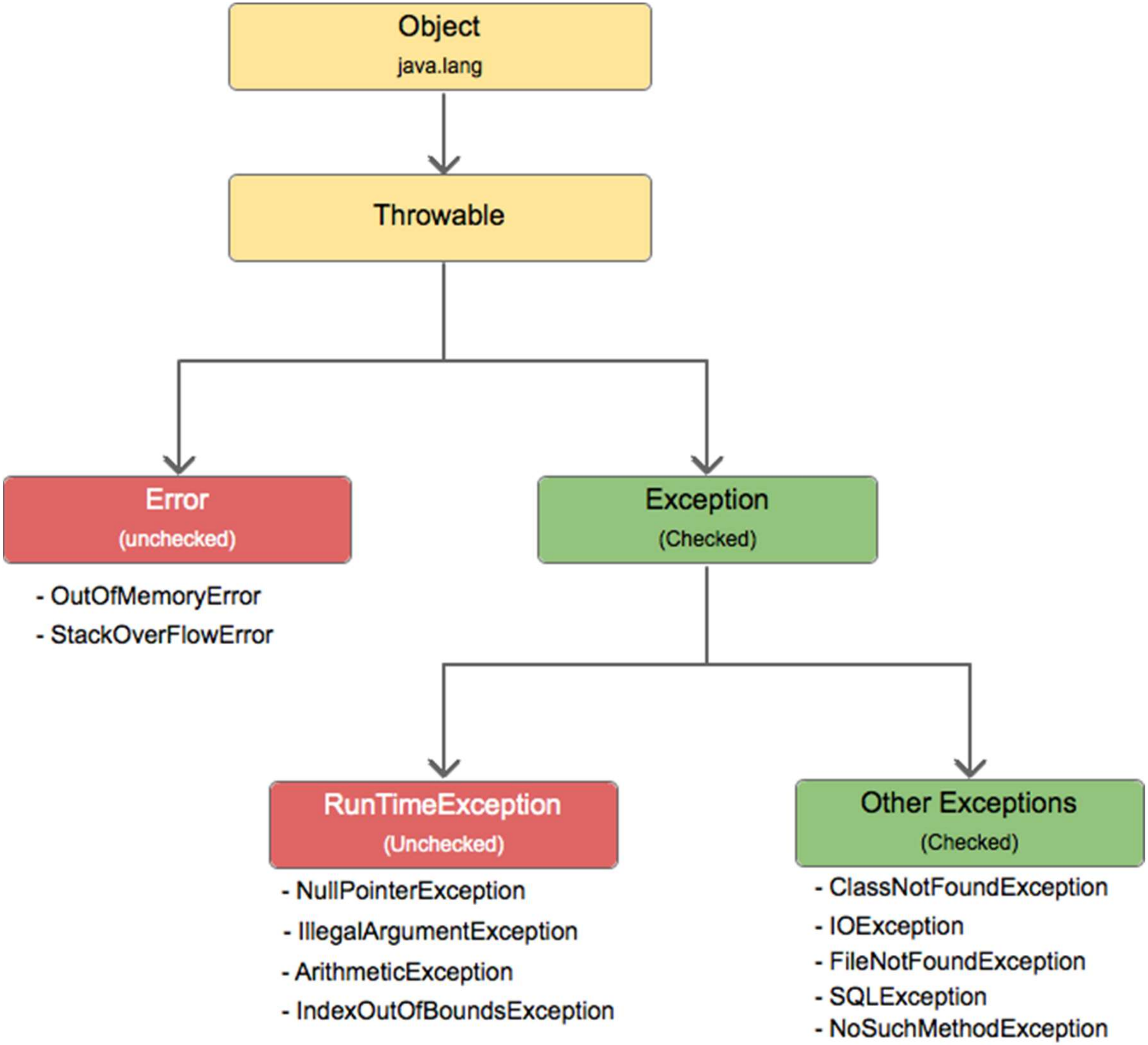
The main advantage is to maintain the normal flow of the program.

Exceptions Handling

```
Statement 1;
Statement 2;
Statement 3;
Statement 4;// Exception occurs
Statement 5;// like divided by zero
Statement 6;
```

The main advantage is to maintain the normal flow of the program.

Hierarchy of Java Exception



Defining Exceptions

Inherit from the `Exception` class.

Has a constructor that takes a `String`.

Defining Exceptions

Inherit from the `Exception` class.

Has a constructor that takes a `String`.

```
public class MyException
  extends Exception {
  public MyException(String message) {
    super(message);
  }
}
```

Example:
`MyException.java`

Throwing Exceptions

Methods must state which exceptions they throw, using the `throws` reserved word

Throwing Exceptions

Methods must state which exceptions they throw, using the **`throws`** reserved word

```
public static void myMethod()
  throws MyException {
  ...
}
```

Throwing Exceptions

Methods must state which exceptions they throw, using the **`throws`** reserved word

```
public static void myMethod()
  throws MyException {
  ...
}
```

```
public static void myMethod()
  throws MyException, OtherException {
  ...
}
```

Throwing Exceptions

Exceptions can be thrown with the **`throw`** reserved word

Throwing Exceptions

Exceptions can be thrown with the **`throw`** reserved word

```
public static void myMethod()
  throws MyException {
  if (...) {
    throw new MyException(“message”);
  }
}
```

Example

- `MyException.java`
- `ThrowMyException.java`

Catching Exceptions

Exceptions can be caught with `try`...`catch`, stopping them from moving up

Catching Exceptions

Exceptions can be caught with `try`...`catch`, stopping them from moving up

```
try {
  myMethod();
} catch (MyException e) {
  System.out.println(e.toString());
}
System.out.println(“GETS HERE”);
```

Example:
`CatchException.java`

5 Keywords

`try:` is used to specify a block where we should place exception code. The `try` block must be followed by either `catch` or `finally`. It means we can not use `try` block alone.

`catch:` is used to handle the `exception`. It must be preceded by `try` block which means we can not use `try` block alone. It can be followed by `finally` block later .

`finally:` is used to execute the important code of the program. It is executed whether an exception is handled or not.

`throw:` is used to throw an exception.

`throws:` is used to declare exceptions. It does not throw an exception. It is always used with method signature and specifies that there may occur an exception in the method.