

# Introduction to Exceptions in Java

1

## Runtime Errors

- What are **syntax errors**?
- What are **runtime errors**?
- Java differentiates between **runtime errors** and **exceptions**
  - Exceptions can be handled in some way
    - Example: division by zero
  - Errors are unrecoverable situations
    - Example: running out of memory

2

## Exceptions

- **Exception** : an abnormal or erroneous situation at runtime
- Examples:
  - Division by zero
  - Array index out of bounds
  - Illegal input number format
  - Following a null reference

3

## Exceptions

- These erroneous situations **throw an exception**
- Exceptions can be thrown by the **runtime environment** or by the **program** itself

4

## Throwing Exceptions

- You have seen a situation where a **program** throws an exception
- Example in ArrayStack.java:

```
public T pop() throws EmptyCollectionException
{
    if (isEmpty())
        throw new EmptyCollectionException("Stack");
    ...
}
```

5

## Java Exceptions

- In Java, an exception is an **object**
  - There are predefined **exception classes** in the Java API
    - **Exception**
      - Its subclass **RuntimeException**
      - Its subclass **NullPointerException**
- etc.

6

## Java Exceptions

- Examples of Java predefined exception classes (types):

`IllegalArgumentException`

`ArrayIndexOutOfBoundsException`

`IOException`

`NullPointerException`

7

## Some Java Error and Exception Classes

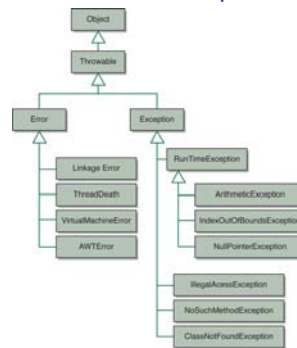


FIGURE 2.7 Part of the Error and Exception class hierarchy

8

## Java Exceptions

- We can throw an exception object of one of these predefined types, or we can define our own exception classes
- How? We can **extend Exception** or one of its subclasses

9

## Example from ArrayStack

```
public class EmptyCollectionException extends RuntimeException
{
    /**
     * Sets up this exception with an appropriate message.
     * @param collection String representing name of collection
     */
    public EmptyCollectionException (String collection)
    {
        super ("The " + collection + " is empty.");
    }
}
```

10

## Uncaught Exceptions

- If an exception is not handled at all by the program, a standard error message is printed by the Java runtime system and the program is terminated
- Example:

```
java.lang.ArrayIndexOutOfBoundsException: 5
at ExceptionExample1.main(ExceptionExample1.java:17)
Exception in thread "main"
```

11

## Catching Exceptions

- To **handle an exception** in your program:
  - A method, or the runtime environment, that detected an error during execution **throws** an exception
  - The code that deals with the exception is said to **catch** or **handle** it

12

## How to Catch an Exception

- Syntax of **try-catch** statement :

```
try
{ // try block: statements(s) that might cause an exception to
  // be thrown
}
catch ( possible-exception-type e )
{ // catch clause: statements to handle the problem,
  // referring to the exception object e
}
```

13

## Catching Exceptions: Example

```
public static void main (String[] args) throws Exception {
    BufferedReader keyboard = new BufferedReader
        (new InputStreamReader(System.in), 1);

    System.out.print("Enter an integer:");

    String userTyped = keyboard.readLine();

    //continued
}
```

14

## Catching Exceptions: Example

```
try {
    int value = Integer.parseInt(userTyped);
}
catch (NumberFormatException e)
{
    System.out.println("Hey, " + e.getMessage() +
        " is not an integer!");
}
}
```

15

## Catching Exceptions

- How **try-catch** works:
  - When the **try-catch** statement is executed, the statements in the **try** block are executed
  - If **no exception** is thrown:
    - Processing continues with the statement following the try-catch statement
  - If an **exception** is thrown:
    - Control is immediately passed to the first **catch** clause whose specified exception corresponds to the class of the exception that was thrown

16

## Catching Exceptions: Example

- Example: try to create a file (in a non-existent directory)

```
String filename = "/nosuchdir/myfilename";
try
{
    new File(filename).createNewFile();
}
catch (IOException e)
{
    System.out.println("Unable to create" + filename + ":" +
        e.getMessage());
}
// execution continues here
```

\* code from Java Developers Almanac

17

## Catching Exceptions: Example

- Here is the output :

```
Unable to create /nosuchdir/myfilename:
The system cannot find the path specified
```

18

## Catching Exceptions

- If an exception is *not* caught and handled where it occurs:
  - Control is immediately returned to the *method that invoked the method* that produced the exception
  - If that method does not handle the exception (via a try statement with an appropriate catch clause) then control returns to the method that called it
- This process is called ***propagating the exception***

19

## Catching Exceptions

- Exception propagation continues until
  - The exception is caught and handled
  - Or until it is propagated out of the main method, resulting in the termination of the program

20