**6. Exceptions**

**The Role of Exceptions**

* An *exception* is Java’s way of saying, “I give up. I don’t know what to do right now. You deal with it.”
* **Understanding Exception Types**



* + Error means something went so horribly wrong that your program should not attempt to **recover** from it.
  + A *runtime exception* is defi ned as the RuntimeException class and its subclasses. Runtime exceptions are also known as *unchecked exceptions*.
  + A *checked exception* includes Exception and all subclasses that do not extend RuntimeException. Checked Exception must be either ***handled or declared****.*
* Types of exceptions



* Using a *try* Statement



* try statements are like methods in that the curly braces are required even if there is only one statement inside the code blocks.
* **Adding a *finally* Block**



* There is one exception, When System.exit is called in the try or catch block, finally does not run.
* **Tricky Example:**

30: public String exceptions() {

31: String result = "";

32: String v = null;

33: try {

34: try {

35: result += "before";

36: v.length();

37: result += "after";

38: } catch (NullPointerException e) {

39: result += "catch";

40: throw new RuntimeException();

41: } finally {

42: result += "finally";

43: throw new Exception();

44: }

45: } catch (Exception e) {

46: result += "done";

47: }

48: return result;

49: }

The correct answer is ***before catch finally done***.

* **Recognizing Common Exception Types**
  + **Runtime Exceptions**

1. **ArithmeticException 🡪** code attempts to divide by zero
2. **ArrayIndexOutOfBoundsException 🡪** code attempts to divide by zero index to access an array.
3. **ClassCastException🡪** when an attempt is made to cast an exception to a subclass of which it is not an instance.
4. **IllegalArgumentException🡪** indicate that a method has been passed an illegal or inappropriate argument.
5. **NullPointerException 🡪** when there is a null reference where an object is required.
6. **NumberFormatException 🡪** when an attempt is made to convert a string to a numeric type but the string doesn’t have an appropriate format.

* **Checked Exceptions**
  + 1. **FileNotFoundException 🡪** Thrown programmatically when code tries to reference a file that does not exist.
    2. **IOException 🡪** Thrown programmatically when there’s a problem reading or writing a file.
    3. ***FileNotFoundException is a subclass of IOException****.*
* **Errors**

1. **ExceptionInInitializerError 🡪** Thrown by the JVM when a static initializer throws an exception and doesn’t handle it.
2. **StackOverflowError 🡪** Thrown by the JVM when a method calls itself too many times (this is called *infinite recursion* because the method typically calls itself without end).
3. **NoClassDefFoundError 🡪** Thrown by the JVM when a class that the code uses is available at compile time but not runtime.

* **Subclasses- Overriding method with exceptions**
  + When a class overrides a method from a superclass or implements a method from an interface, it’s not allowed to add new **checked exceptions** to the method signature. This rule applies only to **checked exceptions.**
* **Printing an Exception**
  + There are three ways to print an exception.
  + **Example:**

5: public static void main(String[] args) {

: try {

7: hop();

: } catch (Exception e) {

9: System.out.println(e);

10: System.out.println(e.getMessage());

11: e.printStackTrace();

12: }

13: }

14: private static void hop() {

15: throw new RuntimeException("cannot hop");

16: }

java.lang.RuntimeException: cannot hop

cannot hop

java.lang.RuntimeException: cannot hop at trycatch.Handling.hop(Handling.java:15)

at trycatch.Handling.main(Handling.java:7)