Object Oriented Software Development

Week 11

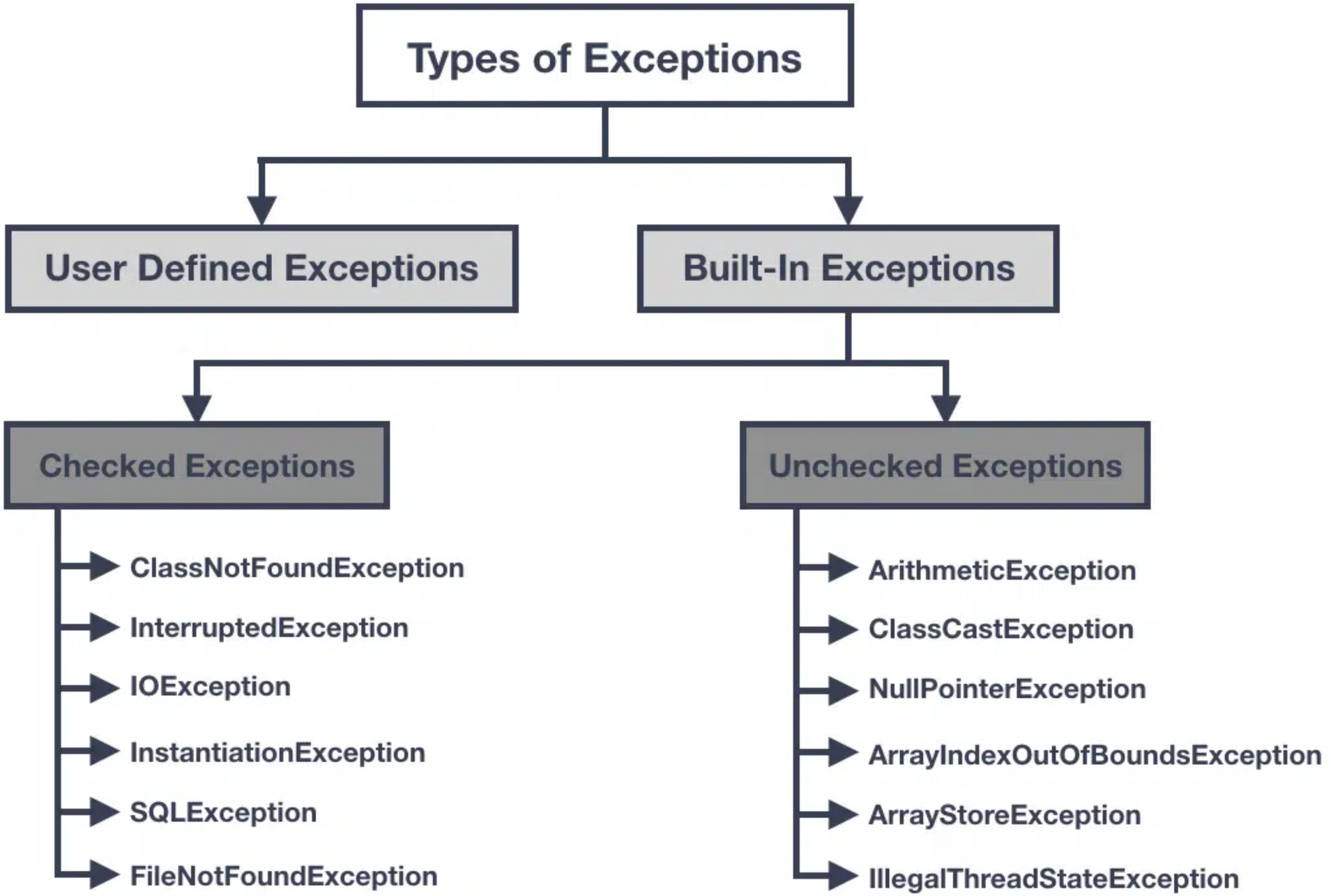
Exceptions

The way to handle run time exceptions in Java

Checked: are detected at compile time, **must** be handled

Unchecked: are not detected at compile, can be ignored or handled

Here, checked/unchecked means are they checked BY THE PROGRAM


Types of Exceptions
User Defined Exceptions
Built-In Exceptions
Checked Exceptions
Unchecked Exceptions
ClassNotFoundException
InterruptedException
IOException
InstantiationException
SQLException
FileNotFoundException
ArithmeticException
ClassCastException
NullPointerException
ArrayIndexOutOfBoundsException
ArrayStoreException
IllegalThreadStateException

Handling Checked Exceptions

throws keyword in method header

```
public static void main(String[] args) throws IOException
    {

        FileReader file = new FileReader("C:\\test\\a.txt");

        BufferedReader fileInput = new BufferedReader(file);

        for (int counter = 0; counter < 3; counter++)
            System.out.println(fileInput.readLine());

         fileInput.close();
    }
```

try-catch-finally

```
class Test
{
    public static void main(String args[])
    {
        System.out.println(Test.test());
    }

    public static String test()
    {
        try {
            System.out.println("try");
            throw new Exception();
        } catch(Exception e) {
            System.out.println("catch");
            return "return";
        } finally {
            System.out.println("finally");
            return "return in finally";
        }
    }
}
```

```
try
catch
finally
return in finally
```

Custom Exceptions

```java
class InvalidAgeException  extends Exception {
    public InvalidAgeException (String str) {
        super(str);
    }
}
```

```java
class TestCustomException1 {
    // method to check the age
    static void validate (int age) throws InvalidAgeException{
        if(age < 18){
            // throw an object of user defined exception
            throw new InvalidAgeException("age is not valid to vote");
        }
        else System.out.println("welcome to vote");
    }
    public static void main(String args[])
    {
        try {
            validate( age: 13);
        }
        catch (InvalidAgeException ex) {
            // printing the message from InvalidAgeException object
            System.out.println("Exception occured: " + ex);
        }
    }
}
```

Output:

Exception occured: InvalidAgeException: age is not valid to vote

Pairs or individually

Share a single device

Javadoc

Everything that is **not private** needs Javadoc

Don't need for getters/setters/constructors

Do need for **classes** e.g. /** Class for the Pacman object */

Do need for **abstract** methods, but **don't** need for each overridden method if it does the same thing (e.g. move())

CISSA

SWOTVAC REVISION WORKSHOP 2023

OBJECT ORIENTED SOFTWARE DEVELOPMENT

SWEN20003

31 MAY 2023
2:00 PM - 4:00 PM
DUAL DELIVERY
ALAN GILBERT G18

UWSU | AFFILIATED CLUB