# Exceptions Practice

Use jshell (part of the JDK) or BlueJ codepad to run these examples and discover the exception. If you use BlueJ you need to perform imports.

Some useful jshell commands:

| | |
|---|---|
| `/help` | - print help. "/help command" for help on a command |
| `/imports` | - list all imported classes and packages |
| `/vars` | - list the variables you have defined and their values |
| `/list` | - list the code you have typed |
| `/exit` | - quit |

```
jshell> /imports
|    import java.io.*
|    import java.math.*
|    import java.net.*
|    import java.nio.file.*
|    import java.util.*
etc.
```

1. What exception is thrown by these statements?

```
double[] score;
score.length
Double[] scores = {1.0, 2.0, 3.0};    // autobox double primitives as double objects
scores[3]
List<Double> list = Arrays.asList(scores);
list.get(list.size())
```

2. Complete this table. Identify each exception as "checked" or "unchecked" (the "Checked?" column). To find out what exception a method throws look in the Java API!

Checked Exception - an exception that you are <u>required</u> to handle using "try - catch" or declare "throws".

Unchecked Exception - an exception that you are <u>not</u> required to handle.

| Exception | Checked? | Example - if blank, give your <u>own</u> example |
|---|---|---|
| NullPointerException | unchecked | |
| ClassCastException | | |
| ArrayIndexOutOfBounds Exception | | |

| | | |
|---|---|---|
| | | `List<String> list = new ArrayList<>( );`<br>`list.add("foo");`<br>`String foo = list.get(2);` |
| | | `int n = 0;`<br>`int m = 1/n;` |
| | | *`// use scanner to parse a String ("four")`*<br>`Scanner scanner = new Scanner("four");`<br>`int n = scanner.nextInt( );` |
| | | `Scanner scanner = new Scanner("four");`<br>`scanner.next()`<br>`scanner.next()` |
| | | `// FileReader reads a file as characters`<br>`// Similar to InputStream (reads bytes)`<br>`FileReader in = new FileReader(`<br>`"doesnotexist.foo");` |
| | | `// create String for money value`<br>`double value = 10.0;`<br>`String s = String.format("`**`%d`**` Baht", value);` |
| | | `// What exception should be thrown?`<br>`Purse purse = new Purse(-1);`<br>`// or here:`<br>`purse.insert( new Coin(-1,"Baht") );` |

3. Find all possible exceptions in this `equals` method. At each line that may throw exception, identify the *type* of exception and the *cause* (what values would cause the exception to be thrown).

```
public class Person {
    private String name;
    private LocalDate birthday;

    /** initialize a new Person object. */
    public Person(String name, LocalDate bday) {
        this.name = name;
        this.birthday = bday;
    }

    public boolean equals(Object obj) {

        Person other = (Person)obj;

        return this.birthday.equals(other.birthday)
               && this.name.equals(other.name);
    }
```

4. We want to ask the user to guess a number, but we don't want the program to crash if user inputs an invalid answer. If he inputs an invalid guess, just print "please input an integer" and *discard the input*. Modify the code so that:

a) add try - catch *only* around the part of code that may throw exception.

b) fix the declaration of "int guess" to avoid scope problem when using try - catch.

c) discard the input line if the guess is invalid. (Actually, it is OK to *always* discard the rest of the input line, in case the use accidentally typed something extra.)

If you don't discard the input line, scanner will get "stuck" at the input it has not read yet, so the same error will occur again.

[You might want to actually do this in your GameConsole so you can check your answer.]

```
public int play(GuessingGame game) {
    Scanner console = new Scanner(System.in);
    while(true) {
        // print a message or hint from game
        System.out.println( game.getMessage() );

        System.out.print("Your guess? ");

        //TODO add try - catch to catch invalid input (not an int)

        int guess = console.nextInt( );


        if (game.guess(guess)) {
             System.out.printf("Right! The secret is %d\n", guess);
             return guess;
        }
     }
}
```

5. Some novice programmers put try - catch around big blocks of code instead of only the part they want to check, and catch <u>every</u> exception instead of <u>specific</u> exceptions, as shown below.

Explain why this can result in inaccurate error messages or "hidden" errors.

```
        System.out.print("Your guess? ");

        try {
            int guess = console.nextInt( );
            if (game.guess(guess)) {
                 System.out.printf("Right! The secret is %d\n", guess);
                 return guess;
            }
        } catch (Exception ex) {
            System.out.println("Sorry, fumble fingers");
        }
```