Tutorial – Week 10 5COSC019W – Object Oriented Programming – Java

<u>Exception Handling and Testing</u>

<u>28-29 November</u>

Exception Handling

1) Try to compile the following code:

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

            int myArray[] = new int[5];
            // trying to access element 5
            System.out.println(myArray[5]);
         }
```

Why do you get the error?
When you try to compile and run the above program, the following message "IndexArrayOutOfBoundsException" is printed to the standard output. This happens because you try to access index 5 that does not exist. Remeber the index in an array can be from 0 to N-1, where N is the length of the array.

2) Add some code to change the previous uncontrolled standard message to the following customized message:

`Message: "The element 5 does not exist!".`

```
public class MyClass{

   public static void main(String args[]){
      try{
         int myArray[] = new int[5];
         // trying to access element 5
         System.out.println(myArray[5]);
      }
      catch(ArrayIndexOutOfBoundsException e){
         System.out.println("The element "+ e.getMessage() + " does not exist!");
      }
   }
}
```

You catch the exception

You handle the error

3) In the code you implemented in your previous tutorial, you loaded an image using `getClass().getResource("zero.png")`.
zero.png has to be saved in the project forlder. What happens if the image is not saved there? In order to reply to this question run the application loading an image that is not in the current path.

You will receive this error: `Exception in thread "main" java.lang.NullPointerException`

Use try and catch to handle the previous exception.

```
try{
      imageIconOne = new ImageIcon(getClass().getResource("one.png"));
      }catch(NullPointerException e){
          System.out.println("Image one not found");
      }
```

4) Consider the following code:

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


        Scanner input = new Scanner(System.in);

        int value = 0;
        System.out.println("Please enter an integer");
        value = input.nextInt();

        System.out.println("Value: " + value);
    }
```

If instead of an integer the user inserts a String an error will be displayed. Modify the code in order to validate the user input and handle the error in case characters are inserted instead of numbers.

JUnit Exercise

5) Using the **Date class** that was provided in tutorial 2 write the method

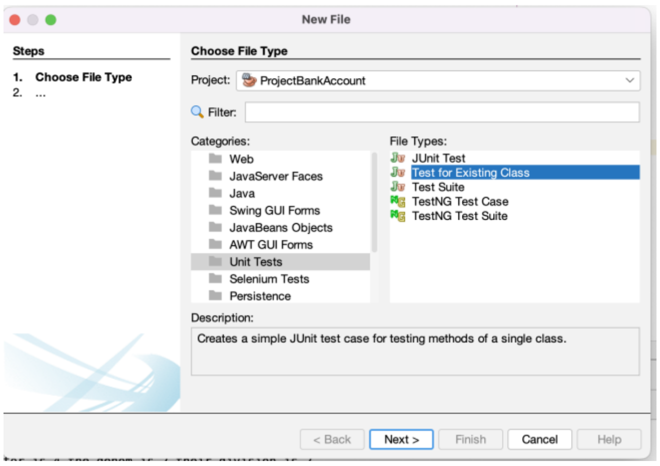
- `public void addDays(int days)    // advances by` *`days`*

Come up with unit tests to check that the addDays method works properly.

<u>Some guidance:</u>

Writing Java and Test Classes Using NetBeans IDE:

1) Right click on the Java project, Select **New -> Other.** Select **Unit Tests** and then **Test for Existing Class**



Steps
1. Choose File Type
2. ...

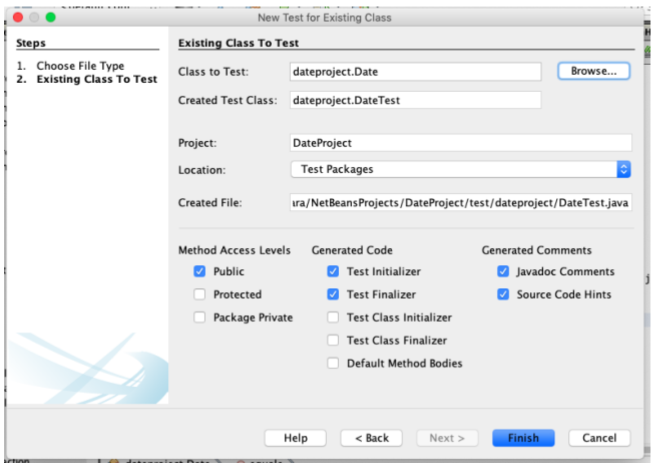
Choose File Type
Project: ProjectBankAccount
Filter:

Categories: Web, JavaServer Faces, Java, Swing GUI Forms, JavaBeans Objects, AWT GUI Forms, Unit Tests, Selenium Tests, Persistence

File Types: JUnit Test, Test for Existing Class, Test Suite, TestNG Test Case, TestNG Test Suite

Description: Creates a simple JUnit test case for testing methods of a single class.

2) Choose the class to test and tick the following boxes



Steps
1. Choose File Type
2. Existing Class To Test

Existing Class To Test
Class to Test: dateproject.Date
Created Test Class: dateproject.DateTest
Project: DateProject
Location: Test Packages
Created File: ıra/NetBeansProjects/DateProject/test/dateproject/DateTest.java

Method Access Levels
- [✓] Public
- [ ] Protected
- [ ] Package Private

Generated Code
- [✓] Test Initializer
- [✓] Test Finalizer
- [ ] Test Class Initializer
- [ ] Test Class Finalizer
- [ ] Default Method Bodies

Generated Comments
- [✓] Javadoc Comments
- [✓] Source Code Hints

The following code in black is generated

```
public class DateTest {

    public DateTest() {
    }

    @Before
    public void setUp() {
    }

    @After
    public void tearDown() {
    }

    /**
     * Test of addDays method, of class Date.
     */
    @Test
    public void testAddDays() {

        // add here your tests
        Date d = new Date(12, 2, 2019);

        d.addDays(4);
        assertEquals(2019, d.getYear());   // expected
        assertEquals(2, d.getMonth());     // value should
        assertEquals(16, d.getDay());      // be at LEFT

    }

}
```

3) Override the `public boolean equals(Object obj){}` method in Date class in order to assert if two Date objects are equal:

```
        Date d = new Date(12, 2, 2019);
        d.addDays(4);

        //expected date
        Date exp = new Date(16, 2, 2019);
        assertEquals(exp, d); // assert if two Date objects are equal (equals
//                               method need to be overriden in Date class!)
```

4) Write a test that checks if the method addDays works as expected if adding days that wrap to the next month (Example: 28/12/2019 ->(add 4 days)->1/01/2020)

5) Add a method `public void printDate()` in the class Date. Write a unit test to verify that the output on the screen is equal to the one expected.

<u>Some guidance</u>:

- System.out.print and System.out.println "print" to the screen. In order to capture the otput you can change that:

```
OutputStream os = new ByteArrayOutputStream();
PrintStream ps = new PrintStream(os);
System.setOut(ps);
```

- If you do this, though, you probably want to change these methods back to "normal" when you are done

```
        PrintStream originalOut = System.out;
                // (code from above)
        System.setOut(originalOut);
```

- The final test will be:

```
@Test
    public void printDate_test(){
        // Prepare to capture output
        PrintStream originalOut = System.out;
        OutputStream os = new ByteArrayOutputStream();
        PrintStream ps = new PrintStream(os);
        System.setOut(ps);

        // define the line separator \n if you use println
        // Line separators are different on different systems!
        // This does not work: assertEquals("("The date is 2019-12-28" n",
      os.toString());
        String separator = System.getProperty("line.separator");

        // perform the test
        Date d = new Date(28, 12, 2019);
        d.printDate();
        assertEquals("The date is 2019-12-28" + separator, os.toString());

        //Restore normal operation
        System.setOut(originalOut);
    }
```

Something More
Testing: typical interview questions

Think how to solve address the following issues:

1) We have the following method used in a chess game:

   boolean canMoveTo(int x, int y)

   x and y are the coordinates of the chess board and it returns whether or not the piece can move to that position. Explain how you would test this method.

2) How would you test an ATM in a distributed banking system?