JAVA

Java is a popular programming language, created in 1995.

It is owned by Oracle, and more than 3 billion devices run Java.

It is used for:

- Mobile applications (specially Android apps)
- Desktop applications
- Web applications
- Web servers and application servers
- Games
- Database connection
- And much, much more

Why use JAVA

- Java works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc.)
- It is one of the most popular programming language in the world
- It has a large demand in the current job market
- It is easy to learn and simple to use
- It is open-source and free
- It is secure, fast and powerful
- It has a huge community support (tens of millions of developers)
- Java is an object oriented language which gives a clear structure to programs and allows code to be reused, lowering development costs
- As Java is close to C++ and C#, it makes it easy for programmers to switch to Java or vice versa

Variables

In Java, there are different types of variables, for example:

- String - stores text, such as "Hello". String values are surrounded by double quotes
- int - stores integers (whole numbers), without decimals, such as 123 or -123
- float - stores floating point numbers, with decimals, such as 19.99 or -19.99
- char - stores single characters, such as 'a' or 'B'. Char values are surrounded by single quotes
- boolean - stores values with two states: true or false

Oops concepts

OOP stands for Object-Oriented Programming.

Procedural programming is about writing procedures or methods that perform operations on the data, while object-oriented programming is about creating objects that contain both data and methods.

Object-oriented programming has several advantages over procedural programming:

- OOP is faster and easier to execute
- OOP provides a clear structure for the programs
- OOP helps to keep the Java code DRY "Don't Repeat Yourself", and makes the code easier to maintain, modify and debug
- OOP makes it possible to create full reusable applications with less code and shorter development time

Encapsulation

The meaning of Encapsulation, is to make sure that "sensitive" data is hidden from users. To achieve this, you must:

- declare class variables/attributes as private

- provide public get and set methods to access and update the value of a private variable

Polymorphism

Polymorphism means "many forms", and it occurs when we have many classes that are related to each other by inheritance

For example, think of a superclass called Animal that has a method called animalSound(). Subclasses of Animals could be Pigs, Cats, Dogs, Birds - And they also have their own implementation of an animal sound

Recursion

Recursion is the technique of making a function call itself. This technique provides a way to break complicated problems down into simple problems which are easier to solve.

Java exceptions

When executing Java code, different errors can occur: coding errors made by the programmer, errors due to wrong input, or other unforeseeable things.

When an error occurs, Java will normally stop and generate an error message. The technical term for this is: Java will throw an exception (throw an error).

Java try and catch

The try statement allows you to define a block of code to be tested for errors while it is being executed.

The catch statement allows you to define a block of code to be executed, if an error occurs in the try block.

The try and catch keywords come in pairs.

Collection in java

The **Collection in Java** is a framework that provides an architecture to store and manipulate the group of objects.

Java Collections can achieve all the operations that you perform on a data such as searching, sorting, insertion, manipulation, and deletion.

Java Collection means a single unit of objects. Java Collection framework provides many interfaces (Set, List, Queue, Deque) and classes (ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet).

Code

```
package com.oracle.javawc.entities.shell;


import java.util.Arrays;

import java.util.List;

import static org.assertj.core.api.Assertions.assertThat;

import org.junit.Before;

import org.junit.Test;

import static org.junit.Assert.*;

import static org.assertj.core.api.Assertions.*;

import static org.assertj.core.util.Arrays.*;
```

```
/**
 *
 * @author lgomes
 */
public class WordCountTest {

    public WordCountTest() {
    }

    @Before
    public void setUp() {
    }

    /**
     * Test of showStatistics method, of class WordCount.
     */
    @Test
    public void testShowStatistics() {
    }

    /**
     * Test of countLines method, of class WordCount.
     */
```

```
@Test
public void testCountLineNull() {
    WordCount wc = new WordCount();
    assertThat(wc.countLines(null)).isEqualTo(0);
}

/**
 * Test of countLines method, of class WordCount.
 */
@Test
public void testCountLineEmptyLines() {
    WordCount wc = new WordCount();
    List<String> lines = Arrays.asList("", "", "");

    assertThat(wc.countLines(lines)).isEqualTo(3);
}

/**
 * Test of countLines method, of class WordCount.
 */
@Test
public void testCountLineWithBlankSpaceLines() {
    WordCount wc = new WordCount();
```

```
        List<String> lines = Arrays.asList(" ", " ", " ");


        assertThat(wc.countLines(lines)).isEqualTo(3);

    }


    /**

     * Test of countLines method, of class WordCount.

     */

    @Test

    public void testCountLine4Lines() {

        WordCount wc = new WordCount();

        List<String> lines = Arrays.asList("line1 line1", "line2 line2", "line3
line3", "line4 line4");


        assertThat(wc.countLines(lines)).isEqualTo(4);

    }


    /**

     * Test of countWords method, of class WordCount.

     */

    @Test

    public void testCountWordsNull() {

        WordCount wc = new WordCount();
    assertThat(wc.countWords(null)).isEqualTo(0);
}

/**
 * Test of countWords method, of class WordCount.
 */
@Test
public void testCountWordsEmptyLines() {
    WordCount wc = new WordCount();
    List<String> lines = Arrays.asList("", "", "");
    assertThat(wc.countWords(lines)).isEqualTo(0);
}

/**
 * Test of countWords method, of class WordCount.
 */
@Test
public void testCountWordsWithBlankSpaceLines() {
    WordCount wc = new WordCount();
    List<String> lines = Arrays.asList(" ", " ", " ");
    assertThat(wc.countWords(lines)).isEqualTo(0);
}
```

```
    /**
     * Test of countWords method, of class WordCount.
     */
    @Test
    public void testCountWordsWith8Words() {
        WordCount wc = new WordCount();
        List<String> lines = Arrays.asList("line1 line1", "line2 line2", "    line3 line3", " line4    line4 ", " ");
        assertThat(wc.countWords(lines)).isEqualTo(8);
    }

    /**
     * Test of countAvgLettersPerWords method, of class WordCount.
     */
    @Test
    public void testCountAvgLettersPerWordsNull() {
        WordCount wc = new WordCount();
        assertThat(wc.countAvgLettersPerWords(null)).isEqualTo(0);
    }

    /**
     * Test of countAvgLettersPerWords method, of class WordCount.
     */
```

```
@Test
public void testCountAvgLettersPerWordsEmptyLines() {
    WordCount wc = new WordCount();
    List<String> lines = Arrays.asList("", "", "");
    assertThat(wc.countAvgLettersPerWords(lines)).isEqualTo(0);
}

/**
 * Test of countAvgLettersPerWords method, of class WordCount.
 */
@Test
public void testCountAvgLettersPerWordsBlankSpaceLines() {
    WordCount wc = new WordCount();
    List<String> lines = Arrays.asList(" ", " ", "   ");
    assertThat(wc.countAvgLettersPerWords(lines)).isEqualTo(0);
}

/**
 * Test of countAvgLettersPerWords method, of class WordCount.
 */
@Test
public void testCountAvgLettersPerWordsSameWordPerLine() {
    WordCount wc = new WordCount();
```

```
        String word = "1234";

        List<String> lines = Arrays.asList(word, word, word);

assertThat(wc.countAvgLettersPerWords(lines)).isEqualTo(word.length());

    }

    /**

     * Test of countAvgLettersPerWords method, of class WordCount.

     */

    @Test

    public                                                          void
testCountAvgLettersPerWordsMultipleLinesAndOneWordPerLine() {

        WordCount wc = new WordCount();

        String word1 = "1234";

        String word2 = "12";

        List<String> lines = Arrays.asList(word1, word1, word1, word2, word2,
word2);

        double expectedResult = (double) (word1.length() + word1.length()

                + word1.length() + word2.length() + word2.length()

                + word2.length()) / 6;

assertThat(wc.countAvgLettersPerWords(lines)).isEqualTo(expectedResult);

    }
    /**
   * Test of countAvgLettersPerWords method, of class WordCount.
   */
  @Test
  public                                                    void
testCountAvgLettersPerWordsMultipleLinesAndTwoWordsPerLine() {
      WordCount wc = new WordCount();
      String word1 = "1234";
      String word2 = "12";
      String line1 = word1+" "+word1;
      String line2 = word2+" "+word2;
      String line3 = word1+" "+word2;
      List<String> lines = Arrays.asList(line1, line2, line3);

      double expectedResult = (double) (word1.length() + word1.length()
              + word2.length() + word2.length() + word1.length()
              + word2.length()) / 6;

assertThat(wc.countAvgLettersPerWords(lines)).isEqualTo(expectedResult);
  }
```

```
    /**
   * Test of countAvgLettersPerWords method, of class WordCount.
   */
  @Test
  public                                                    void
testCountAvgLettersPerWordsMultipleLinesAndMultipleWordsPerLineNoBlank
Space() {
      WordCount wc = new WordCount();
      String word1 = "1234";
      String word2 = "12";
      String word3 = "12345";


      String line1 = word1+" "+word2;
      String line2 = word2+" "+word3;
      String line3 = word1+" "+word3;
      List<String> lines = Arrays.asList(line1,line2,line3);


      double expectedResult = (double) (word1.length() + word2.length()
              + word2.length() + word3.length() + word1.length()
              + word3.length()) / 6;

assertThat(wc.countAvgLettersPerWords(lines)).isEqualTo(expectedResult);
  }
```

```
    /**

 * Test of countAvgLettersPerWords method, of class WordCount.

 */

@Test

public                                                              void
testCountAvgLettersPerWordsMultipleLinesAndMultipleWordsPerLine() {

    WordCount wc = new WordCount();

    String word1 = "1234";

    String word2 = "12";

    String word3 = "12345";


    String line1 = word1+" "+word2+" ";

    String line2 = word2+" "+word3;

    String line3 = word1+" "+word3+"     ";

    List<String> lines = Arrays.asList(line1,line2,line3);


    double expectedResult = (double) (word1.length() + word2.length()

            + word2.length() + word3.length() + word1.length()

            + word3.length()) / 6;

assertThat(wc.countAvgLettersPerWords(lines)).isEqualTo(expectedResult);

}
```

```
/**
 * Test of mostCommonLetter method, of class WordCount.
 */
@Test
public void testMostCommonLetterNull() {
    WordCount wc = new WordCount();
    assertThat(wc.mostCommonLetter(null)).isEmpty();
}


    /**
 * Test of mostCommonLetter method, of class WordCount.
 */
@Test
public void testMostCommonLetterEmptyLines() {
    WordCount wc = new WordCount();
    List<String> lines = Arrays.asList("", "", "");
    assertThat(wc.mostCommonLetter(lines)).isEmpty();
}

/**
 * Test of mostCommonLetter method, of class WordCount.
 */
@Test
public void testMostCommonLetterBlankLines() {
    WordCount wc = new WordCount();
    List<String> lines = Arrays.asList(" ", " ", " ");
    assertThat(wc.mostCommonLetter(lines)).isEmpty();
}


    /**
 * Test of mostCommonLetter method, of class WordCount.
 */
@Test
public void testMostCommonLetterUniqueResultOneWordPerLine() {
    WordCount wc = new WordCount();
    List<String> lines = Arrays.asList("1", "22", "333");
    assertThat(wc.mostCommonLetter(lines)).isEqualTo("3");
}


/**
 * Test of mostCommonLetter method, of class WordCount.
 */
```

```
    @Test

    public                                                                  void
testMostCommonLetterDuplicatedResultsWithNumbersOneWordPerLine() {

        WordCount wc = new WordCount();

        List<String> lines = Arrays.asList("111", "222", "333", "aaa", "bbb");

        assertThat(wc.mostCommonLetter(lines)).isEqualTo("1");

    }




    /**

     * Test of mostCommonLetter method, of class WordCount.

     */

    @Test

    public                                                                  void
testMostCommonLetterDuplicatedResultsOnlyLettersSortedLinesOneWordPer
Line() {

        WordCount wc = new WordCount();

        List<String> lines = Arrays.asList("aaa", "bbb", "ccc", "ddd", "zzz");

        assertThat(wc.mostCommonLetter(lines)).isEqualTo("a");

    }




        /**

     * Test of mostCommonLetter method, of class WordCount.
     */
    @Test
    public void testMostCommonLetterUniqueResultMultipleWordsPerLine() {
        WordCount wc = new WordCount();
        List<String> lines = Arrays.asList("1 22 333", "4444 55555", "666666
");
        assertThat(wc.mostCommonLetter(lines)).isEqualTo("6");
    }


    /**
     * Test of mostCommonLetter method, of class WordCount.
     */
    @Test
    public                                                              void
testMostCommonLetterDuplicatedResultsWithNumbersMultipleWordsPerLine()
{
        WordCount wc = new WordCount();
        List<String>  lines  =  Arrays.asList("111  aaa",  "222  bbb",  "333  ccc",
"ddd", "eee");
        assertThat(wc.mostCommonLetter(lines)).isEqualTo("1");
    }


    /**
```

```
     * Test of mostCommonLetter method, of class WordCount.

     */

    @Test

    public void testMostCommonLetterDuplicatedResultsOnlyLettersSortedLinesMultipleWordsPerLine() {

        WordCount wc = new WordCount();

        List<String> lines = Arrays.asList("aaa bbb ccc", "ddd eee", " zzz fff ", "ggg", "hhh yyy ooo ");

        assertThat(wc.mostCommonLetter(lines)).isEqualTo("a");

    }




        /**

     * Test of mostCommonLetter method, of class WordCount.

     */

    @Test

    public void testMostCommonLetterWithWordString() {

        WordCount wc = new WordCount();

        List<String> lines = Arrays.asList("word1 word2 word3", "word1 word2 word3");

        assertThat(wc.mostCommonLetter(lines)).isEqualTo("d");

    }



}
```