Assignment 3.1

Eric Wallace

Exercise6.1- Did it

Exercise6.2- They are a standardized format. Common sections include Class Name, Syntax, Fields, Constructors, and Methods. They manipulate the field data.

Exercise6.3- The two versions are startsWith(String prefix) and the overloaded version startsWith(String prefix, int toffset) checks substrings against each the supplied substring.

Exercise6.4- The endsWith takes a String as parameter and returns a boolean for what the given substring ends with.

Exercise6.5- length() takes no parameters and returns int number of characters.

Exercise6.6- I found Oracle's Java Docs to be useful and googled things as well.

Exervise6.7-

String text = "Text to trim";

System.out.println(text.trim());

Exercis6.8- Added .trim() and tested.

Exercise6.9- Added .toLowerCase() method and tested.

Exercise6.10- the return type is boolean.

Exercise6.12- Random is in the java.util package. It generates pseudo-random numbers. It generates by typing:

Random randomNumber = new Random();

then calling nextInt()method:

int randomint = randomNumber.nextInt();

Exercise6.13-

Random randomNumber = new Random();

System.out.println(randomNumber.nextInt());

Exercise6.14- Did it.

Exercise6.15- between 0 and 100.

Exercise6.16- Did it.

Exercise6.17- Added getResonse() method, works like a magic 8 ball.

Exercise6.18- Switched to ArrayList.

Exercise6.20- Did it.

Exercise6.21- It can be used in place of Random. Random helps security be more impenetrable by being less predictable.

Exercise6.22- Did it.

Exercise6.23- As more are added, they become less repetitive and more random.

Exercise6.24- HashMap is an unordered collection that stores values in key/value pairs.

Exercise6.25- Functions that depend on Hashmap are compute, computeIfPresent, computeIfAbsent, forEach, getOrDefault, merge, put, replace and replaceAll.

Exercise6.26- Using the .size() method.

Exercise6.28- It will overwrite the key/value.

Exercise6.29- It adds two entries into the HashMap.

Exercise6.30- map.containsKey(String key);

Exercise6.31- It returns null.

Exercise6.32- keySet() method.

Exercise6.33- Did it.

Exercise6.34- ArrayLists are ordered and use the get(index) method to return the value. HashSets are not ordered, you can't have duplicate entries and there is no get() method.

Exercise6.35- The split method can be used on a single or multiple delimiter(s) such as splitting on empty space so that all the words are separated or splitting on punctuation. They can be used on a substrings.

Exercise6.36-

sentance.split(" ");

sentance.split(" t");

Exercise6.37- HashSets will not have duplicates and ArrayLists are ordered.

Exercise6.38- If there is more than one space the consecutive spaces are saved as additional empty strings.

Exercise6.39- Adding static to the method means it is able to be called in another method using the . before it. This will not create an object but still be accessed on the class name.

Exercise6.40- Array's have searching, comparing, copying, fill, and sorting methods.

Exercise6.41- Did it.

Exercise6.42- Did it.

Exercise6.43- Did it.

Exercise6.44- Did it.

Exercise6.47- Keep getting errors.

Exercise6.48- The putIfAbsent method checks whether theres a value associated with the key. If not it will associate the input to the value, if the key is taken if returns the current value.

Exercise6.49- Did it.

Exercise6.50- Did it.

Exercise6.51- It looks accurate to me. The descriptions could be better. I don't see errors.

Exercise6.52- They make the code and formatting easier to read and informs the reader important pieces to the code.

Exercise6.53-

@code signals to javadoc not to treat the following code as HTML but as a code sample instead.

@ deprecated designates code that shouldn't be used anymore.

@exception or @throws add a Throws subheading to the generated documentation along with the name and description.

@see acts as a reference designator.

Exercise6.54- Did it.

Exercise6.55- The drawSquare method creates a blue square, the drawWheel method draws a red wheel caricature and the drawScribble draws random colored scribbles.

Exercise6.56- I tested a few of the methods.

Exercise6.57- Did it, tested the methods of canvas.

Exercise6.58- All but square and clear methods have color constants.

Exercise6.59- Color.GRAY, Color.gray, Color.CYAN, Color.DARK\_GRAY.

Exercise6.61- .erase()

Exercise6.62- Did it.

Exercise6.63- Did it.

Exercise6.64- Dit it.

Exercise6.65- Did it.

Exercise6.66- There are 6 methods.

Exercise6.67- Did it.

Exercise6.68- Did it.

Exercise6.69- HashSet made the most sense, it stores unique items.

Exercise6.70- Did it.

Exercise6.71- Did it.

Exercise6.72- Did it.

Exercise6.73- The lower the number the slower the ball will fall, higher falls faster.

Exercise6.74-

public static final double TOLERANCE = 0.001;

private static final int PASS\_MARK = 40;

public static final char HELP\_CMD = 'h';

Exercise6.75- E and PI

Exercise6.76- It keeps with the concept of DRY or don't repeat yourself. The code is easier to read that way.

Exercise6.77-

public static int max(int a, int b)

Exercise6.78- Math objects are not supposed to be instantiated. Because of this they can't be instance methods.

Exercise6.79-

public class TestCountTIme

{

public long start;

public long time;

public TestCountTime()

{

public void test(){

start = System.currentTimeMillis();

for (int i = 0; i < 10000; i++)

System.out.print((i + 1) + " ");

time = system.currentTimeMillis() - start;

System.out.println("\nThe time it takes to count " + "from 1 to 10000 is " + time + "miliseconds.");

}

}

}

Exercise6.80- Yes. No.

public class MethodTester

{

public MethodTester()

{

}

public static void staticMethod()

{

System.out.println("Static method");

}

public void instanceMethod()

{

System.out.println("Instance method");

staticMethod();

}

public static void anotherStaticMethod()

{

System.out.println("Another static method");

staticMethod();

}

}

Exercise6.81- Did it.

Exercise6.82- DId it.

Exercise6.83- I exported a jar file and updated Java, still couldn't get it to run.

Exercise6.84-

public class TestConstructionCount

{

private static int constructionCount = 0;

public TestConstructionCount()

{

constructionCount++;

}

public static int numberOfInstances()

{

return construcitonCount;

}

}

Exercise6.85- Did it.

Exercise6.87-

public class NameGenerator

{

public NameGenerator()

{}

public static String generateStarWarsName(String firstName, String lastName, String mothersMaidenName, String hometown)

{

String swFirstName = lastName.substring(0, 3) + firstName.substring(0, 2);

String swLastName = mothersMaidenName.substring(0, 2) + hometown.substring(0, 3);

return swFirstName + " " + swLastName;

}

}

Exercise 6.88-

public void printUpper(String s)

{

s = s.toUpperCase();

System.out.println(s);

}

Exercise6.89- You can't pass primitives or objects to methods as parameters you can only pass the value. It has to be done through the code.