LAB ASSIGNMENT III

Instructions

- This assignment is a continuation of the second assignment, complete the second assignment to start with this.
- This assignment must be done with ***eclipse IDE***, go through the ***eclipse IDE*** *handout* if you find difficulties.
- Do utilize the JDK documentation and e-books for any doubts.

Step 7: Static methods

- Static methods (as well as static attributes) belong to a class rather than an object. It is invoked using the class name and not the object instance.
- The add method above can also be implemented in the following way.
- Here both the objects is passed explicitly because the `add()` is not a part of any object.

```
--------------Fraction.java --------------

public static Fraction add (Fraction first, Fraction second) {
      int numerSum;
      if ( first.getDenominator() == second.getDenominator() ) {
        numerSum = first.getNumerator() + second.getNumerator()
        return ( new Fraction(numerSum, first.getDenominator()) );
      }
      else {
        // Your code to implement in this case
      }
  }
```

- The driver code is as follows

```
Fraction f1 = new Fraction(5,8);
Fraction f2 = new Fraction(2,8);
Fraction f3 = Fraction.add(f1,f2); // Note 'Fraction.add()'
f3.print();
```

- Implement a `isEqual()` method to check whether two fractions are equal.

```
public boolean isEqual(Fraction Second){
   // Write your code here
```

```
}
```

- It is not hard to improve the `add()` method to compute sum even if denominators are different. Try to implement. Make use of the existing methods like `multiply()` to achieve this.
- Conversion of fractions to same base is required for subtraction too. Hence it is a good idea to implement a separate `convert()` method that can be used by both `add()` and `subtract()`.
- Note: It is a good programming practice to avoid using static methods. It sort of kills the spirit of object oriented programming.

Step 8: Static variables (or attributes)

- Like static methods, static variables belong to a class not to any particular object instance.
- Static variables can only be accessed by static methods. For example, consider the class.
- Let's define a static variable '`count`' to keep track of number of Fraction objects created.
- We also define a static method `incrementCount()` which is invoked in each constructor.

  Note: The constructor is called each time an object is instantiated. So count is incremented.

```
public class Fraction {
  private int numerator;
  private int denominator;
  private static int count = 0;

  Fraction() {  // default constructor
      this.numerator = 1;
      this.denominator = 1;
      incrementCount();
  }
  Fraction(int n) { // Another constructor
      this.numerator = n;
      this.denominator = 1;
      incrementCount();  }
```

```
  Fraction(int n, int d) {
    this.numerator = n;
    this.denominator = d;
    incrementCount();
  }

    ..................
    ..................

  public static void incrementCount() {
    count++;
  }

  public static void printCount() {
     System.out.println("count = " + count);
  }
}
```

- The driver code as follows

```
Fraction.printCount();  // Should print 0
Fraction f1 = new Fraction(2,3);
Fraction.printCount();
Fraction f2 = new Fraction();
Fraction.printCount();
```

- Since a 'static' attribute belongs to the class, you cannot initialize it using constructor. The value has to be assigned while the attribute is declared itself.
- Since `incrementCount()` needs to be called only when a constructor is called, it is safer to make it private so that it cannot be called from outside (i.e. Driver).

```
private static void incrementCount() {
    count++;
  }
```

Step 9: Working with Array of Objects

- An array is a container object that holds a fixed number of values of a single type
- The length of an array is established when an array is created. After creation its length is fixed.
- We will now create an array of fractions in Driver5.java.

```
Fraction[] fracArray = new Fraction[5];
```

- This creates a single array object that can hold/contain 5 Fraction objects.

- Each of the 5 Fraction object has to be created explicitly.

```
fracArray[0] = new Fraction(2,5);  // creates the first fraction
fracArray[2] = new Fraction();    // creates the third fraction (1/1
                                     set by default)
fracArray[3] = new Fraction(4);  // creates the 4th fraction (4/1 is
                                     set)
```

- The methods of these objects can be called to manipulate the fractions. For example

```
fracArray[0].inverse().print();  // prints 5/2
fracArray[2].multiply(fracArray[3]).print()  // prints 4
```

- Try calling `fracArray[1].print();  // Check what happens – Object not created`
- Try calling `fracArray[7].print(); // Check what happens – Out of bounds`
- Try `System.out.println(fracArray.length);  // What happens`
- Loop through the array to assign, manipulate and print the fractions. Get comfortable working with arrays

Note: **Arrays of primitive data types**

- For working with arrays of primitive data type creation of array object is enough. For example.

```
int[] intArray = new int[5];
intArray[0] = 100;
intArray[1] = 200;
      :
      :
```

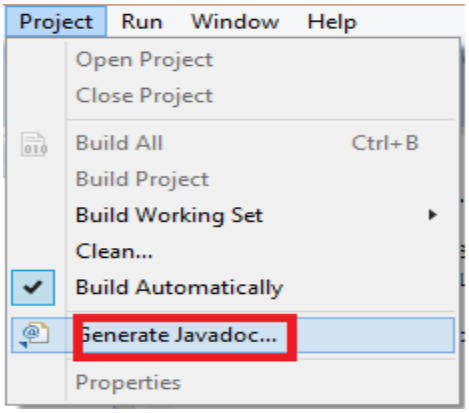
- Primitive data types: `byte, short, int, float, double, boolean, char, String`
- String is an object but treated as primitive for all practical purposes. It will be discussed later.
- Direct initialization of primitive data types is also possible

```
float[] floatArray = {2.8, 3.5, 15.6, 7.99}; // floatArray.length is
                                                automatically set to 4
char[] charArray = {'a', 'b', 'c'};  // charArray.length is set 3
```

Step 10: Generating documentation using Eclipse/ Terminal

Using Eclipse IDE

1. Open the Eclipse project folder.
2. Select Project > Generate Javadoc....



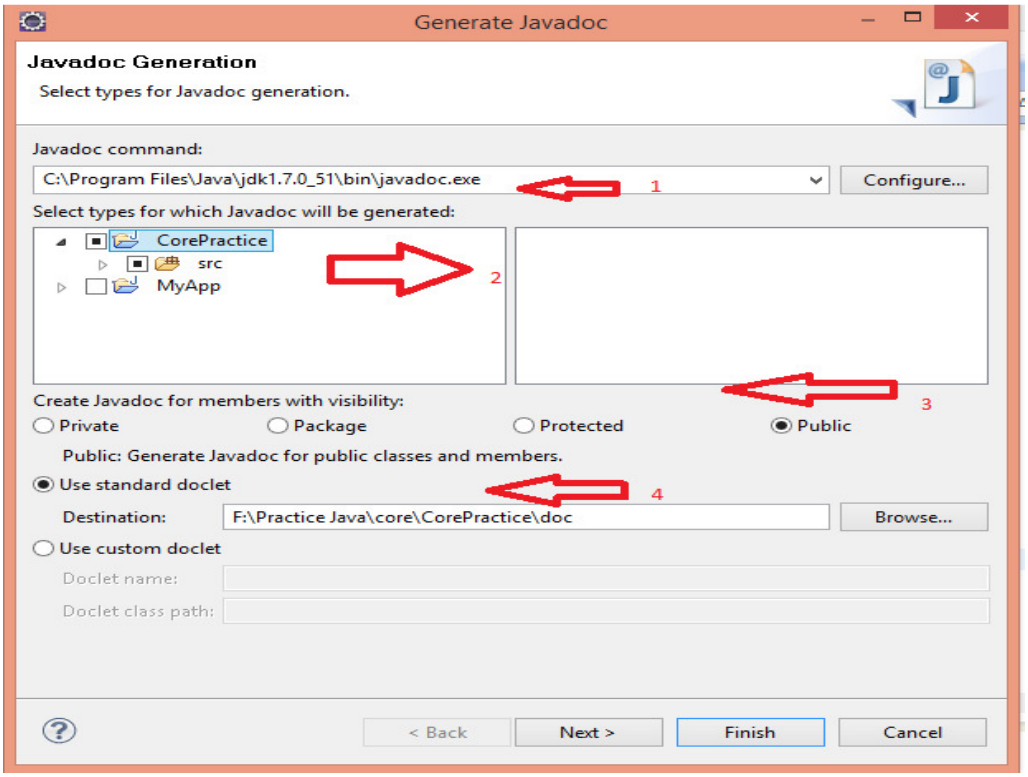
3. At first step of the wizard, you can define settings for:

   3.1. Select path for the *javadoc.exe* tool from the JDK.

   3.2. Project resources for which to generate Javadoc.

   3.3. Classes and methods for which to generate Javadoc based on their visibility.

   3.4. Location of the Javadoc (by default it will be placed in the *doc* folder in the project location).



- Open Fraction.html in a browser. It contains the necessary information for an user to know what all functionalities are supported by Fraction so that they can use it.
- You don't have to write a separate manual stating what functionalities are supported by your class. It is automatically created by javadoc.
- Take your time to understand what is in the html page.
- Are private attributes or methods mentioned in the html? Now change an attribute to public.

```
public int numerator;
```

- Run javadoc, regenerate Fraction.html and check.
- Now in Fraction.java, just above the add method include a comment using `/** ... */`

```
/** This method adds two fractions and returns the sum */
public Fraction add(Fraction frac) {
    // Your code
}
```

- Run javadoc, regenerate Fraction.html and check. Adding comments help!!

Using Terminal

- You can generate the documentation for the Fraction class by issuing the following command.

```
CMD> javadoc -d docs Fraction.java // Generates lot of files including Fraction.html in docs folder
```

- For more features of javadoc and its usage check out the web