Assignment 5

Due Sunday Nov 30, 2014 at midnight

Read the instructions below carefully. The instructions must be followed. The assignment is worth 5% of your final grade. No late assignments will be accepted.

Question 1 should be solved in a file called `A5Q1.java`. Question 2 should be solved in a file called `Rectangle.java` and a file `Canvas.java`. Place all the .java files into a folder called `A5_xxxxxx`, where you should replace xxxxxx by your student number. Compress this folder into `A5_xxxxxx.zip` file. Submit your assignment, i.e., `A5_xxxxxx.zip` file, via Blackboard Learn (as instructed in the first lab.)

The questions that do not compile will be graded with mark 0. For that reason, if you run out of time and/or one of your answers contains code that does not compile, then comment out that section of the code.

Question 1: Recursion

Implement a **recursive** Java method to calculate the sum of all digits of a number. Your method must have the following header:

```
public static int digitSum( int n )
```

Then, implement a **recursive** Java method to compute the digital root of a number. Your method must use the `digitSum` method. Your method must have the following header:

```
public static int digitalRoot( int n )
```

The digital root of a number is calculated by taking the sum of all of the digits in a number, and repeating the process with the resulting sum until only a single digit remains. For example, if you start with 1969, you must first add 1+9+6+9 to get 25. Since the value 25 has more than a single digit, you must repeat the operation to obtain 7 as a final answer.

Your method for computing the digital root must use the `digitSum` method. Both methods should be in the same file `A5Q1.java`. For testing, implement a main method that first prompts the user to enter positive integer, second it displays the sum of its digits and third it displays its digital root.
Also test your methods using Junit tests from the provided file `A5Q1Test.java`. You will need to use the Test menu button in DrJava instead of the Run button to execute the tests. You can do more testing by yourself, but make sure that the tests from `A5Q1Test.java` (without modification) execute properly.

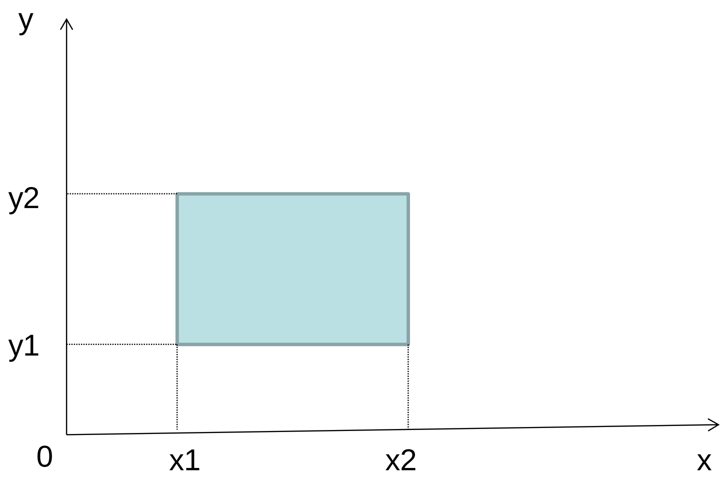
(To do Junit testing, load your `A5Q1.java` solution and the provided `A5Q1Test.java` to Dr.Java. Compile both files, and then press “Test” menu button in Dr.Java. `A5Q1Test.java` contains 5 tests. Those tests for which your solution computes the correct answer will be displayed in green and those for which your solution fails will be displayed in red. Make sure all the provided tests are displayed in green i.e. that they pass).

Note: You can implement iterative versions of the two methods for yourself, but submit the recursive versions only. **Your recursive methods must not use loops.**

Question 2: Objects

Part (a): In this question, you will create a class `Rectangle` that will store information about 2D rectangles that a user is drawing on a computer screen. Place your solution in `Rectangle.java` file. See the UML diagram for the class `Rectangle` for complete information about the class. Your implementation of Rectangle class in Rectangle.java must match the given UML diagram.
(UML diagrams specify the attributes of a class and their types and the method headers that you must use in your Java code, including the types of their parameters and the return types. + means public and – means private. Something underlined should be static.)

Rectangle
- x1: int - y1: int - x2 : int - x2 : int - colour : String
+ Rectangle(x1: int, y1:int, x2:int, y2:int, colour : String) + getColour () : String + setColour(newColour : String) + getX1() : int + getY1() : int + getX2() : int + getY2() : int + move(x1:int , y1: int) + print() + getArea() : int + getPerimeter() : int + contains(x: int, y:int) : boolean



Attributes:
`Rectangle` objects will store the following information:

- The coordinates, x1 and y1, indicating the position of the lower left corner and the coordinates x2 and y2 for the upper right corner. See the above figure for an example. (You may assume that your program will be tested with values x2>=x1 and y2>=y1).

- The colour of the rectangle, as a String such as “red”, “green”, “yellow”, etc. Use lowercase in the strings.

Constructor:
To create a `Rectangle` object, values should be supplied for the coordinates and for the colour. Use the following constructor header:

- `public Rectangle(int x1, int y1, int x2, int y2, String colour)`

Accessors and Modifiers:
Public accessor methods should be available for each coordinate and for the colour. A public modifier method should be available for the colour. Use the following method headers:

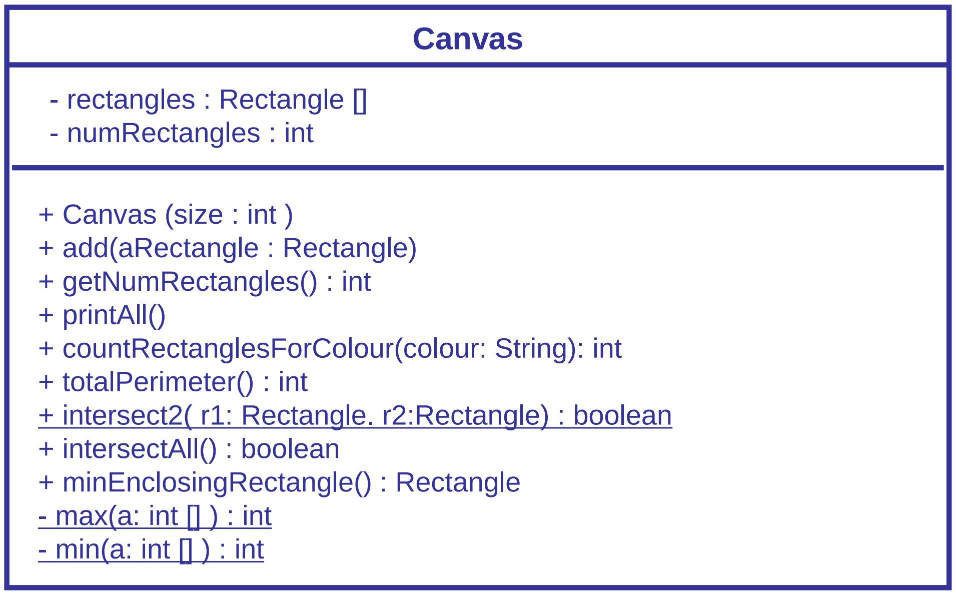
- `public int getX1()` returns the value of x1.
- `public int getY1()` returns the value of y1.
- `public int getX2()` returns the value of x2.
- `public int getY2()` returns the value of y2.
- `public String getColour()` returns the colour.
- `public void setColour (String newColour)` changes the colour.

Additional methods

- `public void move(int x1, int y1)` moves a rectangle so that the new position of the lower left corner becomes x1, y1 while preserving it initial size.
- `public void print()` prints the coordinates and the colour of the rectangle. Example of printed message:
  ```
  The coordinates are (20,30) (60,70)
  The colour is blue
  ```
- `public int getPerimeter()` returns the perimeter of the rectangle.
- `public int getArea()` returns the area of the rectangle.
- `public boolean contains(int x, int y)` returns true if the point with coordinates x, y is inside the rectangle and false otherwise. (A point on the boundary of the rectangle is considered to be inside).

Testing
Test your Rectangle class using the JUnit class `RectangleTest.java`. Load `RectangleTest.java` and your `Rectangle.java` into Dr.Java, compile and use the Test menu button instead of the Run button. You can do more testing by yourself, but make sure that the tests from `RectangleTest.java` (without modification) execute properly.
You can also test by using the provided file `A5Q2.java`, and running the main method. Make sure you load that file before pressing Run (otherwise you will get the message there is no main method, since the other files do not have and do not need a main method). In the file `A5Q2.java`, comment out the last part until you implement part b of question 2. Make sure the output of the first part of `A5Q2.java` is similar to the first part of the provided file `A5Q2.txt`.

Part (b): In this question, you will create a class `Canvas` that will store a collection of `Rectangle` objects, and allow several operations on the collection. See the below UML diagram for the class Canvas for all the information about the class (and for the return types of the methods). Your implementation of Canvas class in Canvas.java must match the given UML diagram.



Canvas

- rectangles : Rectangle []
- numRectangles : int

```
+ Canvas (size : int )
+ add(aRectangle : Rectangle)
+ getNumRectangles() : int
+ printAll()
+ countRectanglesForColour(colour: String): int
+ totalPerimeter() : int
+ intersect2( r1: Rectangle. r2:Rectangle) : boolean
+ intersectAll() : boolean
+ minEnclosingRectangle() : Rectangle
- max(a: int [] ) : int
- min(a: int [] ) : int
```

To model the collection, the class Canvas uses the following two **attributes**: a reference variable to to an array of `Rectangle` objects, and a variable `numRectangles` that will be used to keep track of how many rectangles were actually added.

Constructor: The constructor takes one parameter, the maximum number of Rectangles allowed in the array. It should allocate memory for the specified number of `Rectangle` objects to be stored in the array. However, no `Rectangle` objects are added at this time, so `numRectangles` should be initialized with zero.

Methods: The method `add(Rectangle aRectangle)` will attempt to add the specified `Rectangle` object to the `Canvas`. If there is space to hold another `Rectangle` object, the object should be added. If the `Canvas` has no more room to store additional objects, an error message should be printed. Sample error message:

```
The canvas is full. Unable to add a new rectangle.
```

The method `getNumRectangles()` returns the current number of rectangles in the array. It is an accessor method.

The method `printAll()` prints the information about all the rectangles using the print method from the class rectangles on each element in the array.

The method `countRectanglesForColour(String colour)` goes through the array and counts how many rectangles have the specified colour.

The method `totalPerimeter()` returns the sum of the perimeter values of all the rectangles in the array.

The method `intersect2(Rectangle r1, Rectangle r2)` checks if the two given rectangles intersect (overlap). It is a *static* method because it can be used independent of any object of type

Canvas and it returns true if there is overlap and false otherwise. Definition: two rectangles intersect if they have at least one point in common, otherwise they do not intersect.

The method `intersectAll()` returns true if there exists a point that intersects all rectangles. To test this, it is enough to test if every pair of rectangles intersects (according to a Helly's theorem http://en.wikipedia.org/wiki/Helly's_theorem).

The method `minEnclosingRectangle()` calculates the minimum enclosing rectangle of all rectangles in the canvas. It returns an object of type Rectangle, with the calculated coordinates and any colour you prefer. To find minimum enclosing rectangle you will need to find the minimum x coordinate of all rectangles, the maximum x coordinate for all rectangles, the minimum y coordinate and the maximum y coordinate of all rectangles. Use the private static methods max and min. The former returns the maximum of an array of integers, and the latter the minimum. They are static because they can be used independent of any object of type Canvas. They are private because they are helper methods that should not be used outside the class.

Testing

Test your Canvas class using the class `A5Q2.java`. This should produce an output similar to that produced by the last part of the provided file `A5Q2.txt`. Make sure that class `A5Q2.java` runs without modification. Also do Junit testing using the provided test cases `CanvasTest.java`. You can do more testing by yourself, but make sure that the tests from `CanvasTest.java` (without modification) execute properly.