CS: 113 Introduction to Object Oriented Programming (IOOP) Fall, 2022

Project 2 (100 points) Due 10/24/2022

Submit your solutions to canvas. For programming assignments do not send the entire project. All I want are the files ending in .java. Each class will have to be defined in its own separate .java file. All solutions should be put in one .java file. Please make sure your name is included at the top of each .java file. **There will be no re-submission of work. There will be no exceptions. So, before you submit your work please ensure you have everything the way you want it.**

The grading breakdown for this project is as follows:

10% Readability – Is the program easy to read and understand (indentation, documentation, good use of white space, good output format, user prompts)

10% C++ – Does the program make good use of the C++ constructs (functions, control flow, etc.)

30% Robustness - Does the program compile and run, and not crash or throw exceptions

50% Correctness – Does the program solve the intended problem and work on a variety of reasonable inputs

Problem 1 (30 points)

Implement a `RandomInteger` class.

It should have the following data members:

```
private int min;
private int max;
private int range;
```

It should have the following constructors:

```
public RandomInteger()
public RandomInteger(int min, int max)
```

It should have the following Getters

```
public int GetMin()
public int GetMax()
```

It should have the following Setters

```
public void SetMin(int min)
public void SetMax(int max)
```

Now implement a driver for this class in P1 that uses this class. It can be the guessing game you did previously. It can be the coin flips. It can be the rolling of a die. It can be a new game with randomness that you create.

Problem 2 (30 points)

You have been hired by Industrial Light and Magic to work as a programmer. You are required to design and implement a Java `AnimClock` class. Your class should have a default constructor and a parametric constructor. The constructor should include one parameter. That parameter is a `float` (not a `double`) called `timeStep`. The timeStep value can be no less than a `1/60` of a second, and no greater than `1.0` second. The class should have a method `Tick` which advances total time by the time step. For example, if the time step is `1/10 = .1`, and the method Tick is called 1000 times the total time at that point should be `100.0`. After `100,000` ticks the total time should be `10000.0`. There should be an accessor method GetTime() that returns the total time when it is called. Implement your class, and then implement `P2` to test your class. Do not print out the value after every single tick. Come up with an interval that makes sense. For example, maybe print out total time after every hour. Remember that an hour in simulation time is not real time. So, you should be able to run through 240 hours of simulation time rather quickly. All real numbers should be declared using the type `float`. You can declare `ticks` as a `long`.

Here are some sample runs:

```
Enter the time step
.1
Enter the number of ticks
1000
Enter display interval
100
Time is 10.0
Time is 20.0
Time is 30.0
Time is 40.0
Time is 50.0
Time is 60.0
Time is 70.0
Time is 80.0
Time is 90.0
Time is 100.0

Enter the time step
.1
Enter the number of ticks
```

```
100000
Enter display interval
10000
Time is 1000.0
Time is 2000.0
Time is 3000.0
Time is 4000.0
Time is 5000.0
Time is 6000.0
Time is 7000.0
Time is 8000.0
Time is 9000.0
Time is 10000.0
```

The driver code in `P2` should be exactly like this:

```
Scanner in = new Scanner(System.in);

System.out.println("Enter the time step");
float timeStep = in.nextFloat();

AnimClock clock = new AnimClock(timeStep);

System.out.println("Enter the number of ticks");
long ticks = in.nextLong();

System.out.println("Enter display interval");
long interval = in.nextLong();

for (long i = 1; i <= ticks; i++)
{
    clock.Tick();
    long rem = i % interval;
    if (rem == 0)
    {
       System.out.println("Time is " + clock.GetTime());
    }
}
```

<u>Problem 3 (40 points)</u>

The height, h, of a ball thrown in the air
with an initial velocity of 20 meters from an intial
height of 5 meters can be calculated at t seconds by:

$$h = 5 + 20t + \frac{1}{2}(-9.81)t^2$$

Implement a class called `QuadraticFormula.` Your `P3` driver code should look exactly like this:

```
Scanner in = new Scanner(System.in);
System.out.println("Enter the height of ball");
double h = in.nextDouble();

double a = ?
double b = ?
double c = ?

double[] roots = new double[2];

QuadraticFormula solver = new QuadraticFormula(a, b, c);

Result res = solver.Solve(roots);
switch(res)
{
    case NOT_QUADRATIC:
        System.out.println("Not a quadratic equation");
    break;
    case NO_REAL_SOLS:
        System.out.println("The ball will never reach height " + h);
    break;
    case TWO:
        System.out.println("The ball will reach that height at times " +
                            roots[0] + " and " + roots[1]);
    break;
    case ONE:
        System.out.println("The ball will reach that height at time " + roots[0]);
    break;
 }
```

Given input h, and the equation above you should be able to determine `a`, `b`, and `c` using the standard form for the quadratic equation given by $at^2 + bt + c = 0$. Create all the constructors, and other methods by looking at the above driver code `P3`. At the top of the `QuadraticFormula` you should define the following `enum` type:

```
enum Result {NOT_QUADRATIC, NO_REAL_SOLS, ONE, TWO};
```

Sample output should look like:

```
Enter the height of ball
10
The ball will reach that height at times
0.26755659713779756 and 3.8099153702424267

Enter the height of ball
25.3873598369
The ball will reach that height at time 2.0387355055814957

Enter the height of ball
25.4
The ball will never reach height 25.4
```