SUTD 50.001 Introduction to Information Systems and Programming

Problem set 1

Note: Please submit your answer to TutorJ

Cohort Questions (Week 1)

Session 1

Title: Fibonacci Numbers Generator

1. Write a JAVA program that print out the first 20 numbers in the fibonacci sequence, in this format:

   0,1,1,2,3,5…

   When submitting in TutorJ, return these numbers in this format as a string, instead of printing.

Session 2

Title: Iterating with Iterator

2. Suppose that *integers* is a variable of type List<Integer>. Write a program that uses an iterator to compute the sum of all integer values in the List.
   (Test case inputs: (1, 2, 3, 4, 5) Expected output: 15)

Title: Iterating with For-Each

3. Write a second program that does the same thing as in the previous question but using a for-each loop. (Test case inputs: (1, 2, 3, 4, 5, 6, 7, 8, 9, 10) Expected output: 55)

Session 3

Title: The Account Class

4. Design a class name Account that contains:
   Private int data field named id for the account (default 0)
   Private double data field named balance for the account (default 0)
   Private double data field named annualInterestRate that stores the current interest rate (in percentage, default 0). Assume all accounts have the same interest rate.
   A private Date data field named dateCreated that stores the date when the account was created.
   A no-arg constructor that creates a default account
   A constructor that creates an account with the specified id and initial balance
   The accessor and mutator methods for id, balance, and annualInterestRate
   The accessor method for dateCreated
   A method named getMonthlyInterestRate() that returns the monthly interest rate
   A method named getMonthlyInterest() that returns the monthly interest
   A method named withdraw that withdraws a specified amount from the account
   A method named deposit that deposits a specified amount to the account

Write a test program that creates an Account object, with withdraw and deposit method to withdraw / deposit the amount, and print the balance, monthly interest and the date when the account was created.

Test case

```
public class TestAccount{
  public static void main (String[] args) {
    Account account = new Account(1122, 20000);
    Account.setAnnualInterestRate(4.5);

    account.withdraw(2500);
    account.deposit(3000);
    System.out.println("Balance is " + account.getBalance());
    System.out.println("Monthly interest is " +
      account.getMonthlyInterest());
  }
}
```

Expected output

```
Balance is 20500.0
Monthly interest is 76.875
```

Homework Questions (Week 1)

Title: Birth Year Generator

1. Write a static method, that read in two arguments: current year and your current age, and return your year of birth.
   (Test case input: 2013, 8 Expected output: 2005)

Title: Power2 Generator

2. Write a static method, that print out this list of numbers. (P.S. Don't hardcode, use a for loop). When submitting in TutorJ, return this list of numbers as a string instead of printing.

   1 2 4 8 16 32 64 128

Title: Prime Number Checker

3. Write a static method, that read in a number (you can assume that the input number is always >=3). Return 1 if it is prime, return 0 if it is not prime.

   Hints: use %. a%b= remainder of a/b. e.g. 13%5=3, 4%2=0

(Test case inputs: 4, 7, 14, 23, 99 Expected outputs: 0, 1, 0, 1, 0)

Title: Hailstone Sequence Counter

4. Write a program to generate a hailstone sequence for a given positive integer n and return the number of hailstones in the sequence. Submit your completed program to return the counts for any positive integer n.
   (Test case inputs: 1, 21, 999 Expected outputs: 0, 7, 49)

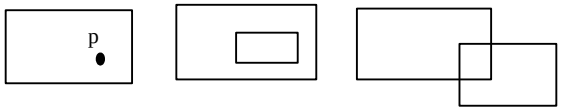
Title: Integer Sorter

5. You are given an unsorted array of up to 10 integers. Write a program to sort the contents of the array in ascending order and return the sorted array.
   (Test case inputs: {2, -3, 1, 5, 0, -10}, {84, 72, 60, 48, 36, 24, 12},{-1, -2, -3, -4, 99, -4, -3, -2, -1}
   Expected outputs: {-10 -3 0 1 2 5}, {12 24 36 48 60 72 84}, {-4 -4 -3 -3 -2 -2 -1 -1 99})

Title: White Space Trimmer

6. Write a method that takes a *List<String>* and removes the leading and trailing white spaces in each list element. Return the list with trimmed elements
   (Test case inputs: [" red ", " white ", " blue "] Expected outputs: ["red", "white", "blue"])

7. Geometry: The MyRectangle2D class
   Define the MyRectangle2D class that contains:
   - Two double data fields named x and y that specify the center of the rectangle with get and set methods: getX, setX, getY, setY. (Assume that the rectangle sides are parallel to x- or y- axes.)
   - The double data fields width and height with get and set methods: getWidth, setWidth, getHeight, setHeight.
   - A no-arg constructor that creates a default rectangle with (0, 0) for (x, y) and 1 for both width and height.
   - A constructor that creates a rectangle with the specified x, y, width, and height: MyRectangle2D(double x, double y, double width, double height)
   - A method getArea() that returns the area of the rectangle.
   - A method getPerimeter() that returns the perimeter of the rectangle.
   - A method contains(double x, double y) that returns true if the specified point (x, y) is inside this rectangle. See Figure 1(a).
   - A method contains(MyRectangle2D r) that returns true if the specified rectangle is inside this rectangle. See Figure 1(b).
   - A method overlaps(MyRectangle2D r) that returns true if the specified rectangle overlaps with this rectangle. See Figure 1(c).



(a) (b) (c)

Figure 1: (a) A point is inside the rectangle. (b) A rectangle is inside another rectangle. (c) A rectangle overlaps another rectangle.

Implement data fields, all constructors, methods getArea(), getPerimeter(), and contains(double x, double y), contains(MyRectangle2D r), and overlaps(MyRectangle2D r)

Please develop test cases to test your code properly before submission.

Cohort Questions (Week 2)

Session 1

1. Please refer to MCQ in TutorJ (Wk.2.1.1 to Wk.2.1.2).

Homework Questions (Week 2)

Title: 2 x 2 Linear Equations

1. Design a class named LinearEquation for a 2x2 system equations of variables x,y:

   ax+by=e

   cx+dy=f

   The class contains:
   Private data fields a, b, c, d, e, and f, the coefficients.
   A constructor with the arguments for a, b, c, d, e, and f.
   Six get methods for a, b, c, d, e, and f.
   A method named isSolvable() that returns true if ad - bc is not 0 (why?)
   Methods getX() and getY() that return the solution for the equation
   Write a test program that prompts user to enter a, b, c, d, e and f and displays the result.
   If ad – bc is 0, report that “The equation has no solution.”

Test case:

```
public class TestLinearEquation {
  public static void main(String[] args) {
    double a = 1.0; double b = 2.0; double c = 3.0;
    double d = 5.0; double e = 6.0; double f = 7.0;

    LinearEquation equation = new LinearEquation(a, b, c, d, e, f);
    if (equation.isSolvable()) {
      System.out.println("x is " +
        equation.getX() + " and y is " + equation.getY());
    }
    else {
      System.out.println("The equation has no solution");
    }

    LinearEquation equation2 = new LinearEquation(1.25, 2.0, 2.0, 4.2, 3.0, 6.0);
    if (equation2.isSolvable()) {
       System.out.println("x is " + equation2.getX() + " y is " + equation2.getY());
    }

    LinearEquation equation3 = new LinearEquation(1.0, 2.0, 2.0, 4.0, 3.0, 6.0);
    System.out.println(equation3.isSolvable());
  }
}
```

Output:

```
x is -16.0 and y is 11.0

x is 0.48000000000000115 y is 1.2

false
```

Title: The Triangle class.

2. Design a class named Triangle that extends GeometricObject. The class contains:
   Three double data fields named side1, side2, side3 with default value 1.0 to denote three sides of the triangle.
   A no-arg constructor to create a default triangle
   A constructor that creates a triangle with the specified side1, side2, and side3
   A method named getArea() that returns the area
   A method named getPerimether() that returns the perimeter
   A method named toString() that returns description of the triangle
   Triangle: side1 = 1.0 side2 = 2.0 side3 = 3.0
   Write a test program to test the code. The program should create the Triangle object with these sides and color and filled properties set.

Test case:

```
public class TestTriangle {
	public static void main(String[] args) {
		Triangle myTri = new Triangle();
		myTri.setColor("red");
		myTri.setFilled(true);
		System.out.println(myTri);
		System.out.println(myTri.isFilled());

		Triangle myTri2 = new Triangle(2.0, 4.0, 5.5);
		System.out.println(myTri2);
		System.out.println("area : " + myTri2.getArea() + " perimeter: " +
myTri2.getPerimeter());

	}
}
```

Output

```
Triangle: side1 = 1.0 side2 = 1.0 side3 = 1.0
true
Triangle: side1 = 2.0 side2 = 4.0 side3 = 5.5
area : 3.0714155938264036 perimeter: 11.5
```

Title: Subclasses of Account

3. In question Week1-S3-CQ4, the Account class was defined to model a bank account. An account has the properties: account id, balance, annual interest rate, and date created, and methods to deposit and withdraw funds. Create a subclass for checking account CheckingAccount. A checking account has an overdraft limit of 5000. Provide constructors for CheckingAccount similar to Account. Override withdraw() to print out "over limit" if the amount withdrawing exceeds the overdraft limit.

Testcase:

```
public class TestCheckingAccount {

   public static void main(String[] args) {

           CheckingAccount myCheckAcc = new CheckingAccount(1024, 8000.0);
           myCheckAcc.deposit(2000);
           myCheckAcc.withdraw(15000);

           System.out.println(myCheckAcc.getBalance());
           myCheckAcc.withdraw(200);
           System.out.println(myCheckAcc.getBalance());
           myCheckAcc.deposit(7000);
           myCheckAcc.withdraw(200);
           System.out.println(myCheckAcc.getBalance());
   }

}
```

Output

```
-5000.0
over limit
-5000.0
1800.0
```

Title: String Operation

4. (Part-I) Design and implement a static method to determine if an input string has all unique characters. Assume the character set is ASCII, which encodes 128 characters into 7-bit binary integers.

http://en.wikipedia.org/wiki/ASCII

Design the most efficient method you can think of, with respect to processing speed.

(Part-II) Design and implement a static method to determine if two input strings are permutation of each other. Assume the character set is ASCII, which encodes 128 characters into 7-bit binary integers.

Design the most efficient method you can think of, with respect to processing speed.

Package your methods into class Pset1.

```
public class Pset1 {
   public static boolean isAllCharacterUnique(String sIn) {
     //todo: add your implementation
   }
   public static boolean isPermutation(String sIn1, String sIn2) {
     //todo: add your implementation
   }
}
```

Test case:

```
public class TestPset1 {
        /**
         * @param args
         */
        public static void main(String[] args) {
                System.out.println(Pset1.isAllCharacterUnique("abcdefghijklmnopqrstuvABC"));
                System.out.println(Pset1.isAllCharacterUnique("abcdefgghijklmnopqrstuvABC"));

                System.out.println(Pset1.isPermutation("@ab", "a@b"));
                System.out.println(Pset1.isPermutation("abcd", "bcdA"));

        }
}
```

Output:

```
true
false
true
false
```

5. Title: Comparable interface

Given the following Octagon class:

```
class Octagon {

        private double side;

        public Octagon(double side){
                this.side = side;
        }

        public double getSide() {
                return side;
        }

}
```

Assume that all eight sides of the Octagon are of equal size.
Modify Octagon class to implement the Comparable<Octagon> interface to allow sorting of Octagon objects based on their perimeters.

Test code:

```
public class Test {
	public static void main(String[] args) {
		ArrayList<Octagon> l = new ArrayList<Octagon>();
		l.add(new Octagon(2));
		l.add(new Octagon(3));
		l.add(new Octagon(1));

		Collections.sort(l);

		for (Octagon o:l)
			System.out.println(o.getSide());

	}
}
```

Results:

```
1.0
2.0
3.0
```