BRAC University
Department of Computer Science and Engineering
Spring 2014
CSE111 (Programming Language-II)

Trace the output of the following Java Codes. Then run them in Dr. Java to see if the results match.

Task 1

//Run the methodA() and methodB() on an Instance of Test few times and explain the answer.

```
public class Test{
int sum;
public int y;
public void methodA(){
int x=0, y =0;
    y = y + 7;
    x = y + 11;
sum = x + y;
System.out.println(x + " " + y+ " " + sum);
  }
public void methodB(){
int x = 0;
    y = y + 11;
    x = x + 33 + y;
sum = sum + x + y;
System.out.println(x + " " + y+ " " + sum);
  }
}
```

Task 2

```
public class Q3
{
public static void main(String args[])
  {
    String test = "";
            int i = 5, j = 0, k = 15;
            while (i< 10){
                        k-=1;
                        j = k;
                        while (j > 10 ){
                                    if (j % 2 == 0){
                                    test = "<--";
                                    test =  test + i + 2 + "-->" + (j / 2);
                                    }
                                    else
                                    {
                                    test = "-->";
                                    test =  "-->" + (i / 2) + test + j;
                                     }
                                    System.out.println(test);
                              --j;
                            }
                        i++;
```

```
                }
        }
```

Task 3

//Run the methodA() on an Instance of Test3 five times and explain the answer.

```
public class Test3{
public int sum;
public int y;

public void methodA(){
        int x=2, y =3;
        int [] msg = new int[1];
        msg[0] = 3;
        y = this.y + msg[0];
        methodB(msg, msg[0]);
        x = this.y + msg[0];
        sum = x + y + msg[0];
        System.out.println(x + " " + y+ " " + sum);
  }

private void methodB(int [] mg2, int mg1){
        int x = 0;
        y = this.y + mg2[0];
        x = x + 33 + mg1;
        sum = sum + x + y;
        mg2[0] = y + mg1;
        mg1 = mg1 + x + 2;
        System.out.println(x + " " + y+ " " + sum);
  }
}
```

Task 4

//Run the methodA() on an Instance of Test4 five times and explain the answer.

```
public class Test4{
public int sum;
public int y;

public void methodA(){
        int x=0, y =0;
        int [] msg = new int[1];
        msg[0] = 5;
        y = y + methodB(msg[0]);
        x = y + methodB(msg, msg[0]);
        sum = x + y + msg[0];
        System.out.println(x + " " + y+ " " + sum);
  }
Private int methodB(int mg2[] , int mg1){
        int x = 0;
        y = y + mg2[0];
        x = x + 33 + mg1;
        sum = sum + x + y;
```

```
        mg2[0] = y + mg1;
        mg1 = mg1 + x + 2;
        System.out.println(x + " " + y+ " " + sum);
        return sum;
  }

private int methodB(int mg1){
int x = 0;
int y = 0;
    y = y + mg1;
    x = x + 33 + mg1;
sum = sum + x + y;
this.y = mg1 + x + 2;
System.out.println(x + " " + y+ " " + sum);
return y;
  }
}
```

Task 5

```
//What is the output if you execute the methodA( ) on an instance of the Test04 Class?

public class Test4{
public int sum;
public int y;
public void methodA(){
int x=0;
int z = 0;
while (z < 5){
      y = y + sum;
      x = y + 1;
System.out.println(x + " " + y+ " " + sum);
sum = sum + methodB(x, y);
z++;
    }
  }
public int methodB(int m, int n){
int x = 0;
int sum = 0;
    y = y + m;
    x = n - 4;
sum = sum + y;
System.out.println(x + " " + y+ " " + sum);
return sum;
  }
}
```

Task 6

```
/*
What is the output for the following code sequence?
FinalT3A fT3A = new FinalT3A();
fT3A.methodA();
fT3A.methodB(6,8);
*/
```

```
public class FinalT3A{
public int sum;
public int y;



public void methodA(){
int x=0, y =0, j = 0;
while (j < 2){
      y = y + j;
      x = j + methodB(y , j);
sum = x + y;
System.out.println(x + " " + y+ " " + sum);
j++;
    }
  }
public int methodB(int p, int k){
int x = 0;
    y = y + k + 1;
    x = x + 3 - p;
sum = sum + x + y;
System.out.println(x + " " + y+ " " + sum);
return sum;
  }
}
```

Task 7

```
class PuzzleTester{
public static void main(String[]args)
{
            Puzzle p = new Puzzle();
            p.methodA();
            p.methodA();
            p=new Puzzle();
            p.methodA();
            p.methodB(7);
   }
}

class Puzzle {
 static int x;
void methodA(){
            int z;
             x=5; //at home, comment/delete this line and try again
             z=x+methodB(x);
            System.out.println(x+" "+z);
             z=methodB(z+2)+x;
            System.out.println(x+" "+z);
            methodB(x,z);
            System.out.println(x+" "+z);
   }
int methodB(int y){
             x=y+x;
            System.out.println(x+" "+y);
```

```
            return x+3;
    }
void methodB(int z, int x){
              z=z+1;
              x=x+1;
            System.out.println(z+" "+x);
    }
}
```

Task 7.1

```
class PuzzleTester{
  public static void main(String[]args)
  {
    Puzzle p = new Puzzle();
    p.methodA();
    p.methodB(7);
  }
}

class Puzzle{
  static int x;
  void methodA(){
    int z;
    x=5; //at home, comment/delete this line and try again
    z=x+methodB(x);
    Maze m1 = new Maze();
    System.out.println(x+" "+z);
    m1.methodA();
    z=methodB(z+2)+x;
    System.out.println(x+" "+z);
    methodB(x,z);
    System.out.println(x+" "+z);
  }
  int methodB(int y){
    x=y+x;
    System.out.println(x+" "+y);
    return x+3;
  }
  void methodB(int z, int x){
    z=z+1;
    x=x+1;
    System.out.println(z+" "+x);
  }
}

class Maze{
  static int x;
  void methodA(){
    int m;
```

```
    x=5;
    m=x+methodB(x);
    System.out.println(x+" "+m);
    m=methodB(m-3)+x;
    System.out.println(x+" "+(m));
    methodB(x,m);
    System.out.println(x+" "+m+x);
  }
  int methodB(int y){
    x=y*y;
    System.out.println(x+" "+y);
    return x+3;
  }
  void methodB(int z, int x){
    z=z-2;
    x=x*1;
    System.out.println(z+" "+x);
  }
}
```

Task 8

Create a class called `Student` as described below:

- **Fields:**
  `name, id, address, cgpa`
- **Methods:**

```
public String getName()
public void setName(String n)
public String getID()
public void setID(String i)
public String getAddress()
public void setAddress(String a)
public double getCGPA()
public void setCGPA(double c)
```

Write a class called `Main` to write a `main()` method:

- `public static void main(String[] args){`

  `}`
- Inside the `main()` method
  - Create 3 objects/instances of `Student` called `john`, `mike` and `carol`
  - Set their fields to some value using the public methods.
  - Print the information of each `Student` using `System.out.println()`

Task 9

Create a class called `BankAccount` as described below:

- **Fields:**
  `name, address, accountID, balance`
- **Methods:**

```
public String getName()
public void setName(String n)
public String getAccountID()
public void setAccountID(String i)
public String getAddress()
public void setAddress(String a)
public double getBalance()
public void setBalance(double c)
public void addInterest() //adds 7% of the balance
```

1. Write a class called `Main` to write a `main()` method:
   - `public static void main(String[] args){`

     `}`
   - Inside the `main()` method
     - Create 3 objects/instances of `BankAccount` called `acc1`, `acc2` and `acc3`
     - Set their fields to some value using the public methods.
     - Call `addInterest()` on `acc1` and `acc3`
     - Print the information of each `BankAccount` using `System.out.println()`

2. Add constructors to `Student` and `BankAccount` and use the constructor to set the field values.

Task 10

Design a class called **`circle`** which contains:

- Two `private` instance variables: `radius` (of the type `double`) and `color` (of the type `String`), with default value of `1.0` and "`red`", respectively.
- Two *overloaded* constructors - a *default* constructor with no argument, and a constructor which takes a double argument for radius.
- Two `public` methods: **`getRadius()`** and **`getArea()`,** which return the radius and area of this instance, respectively.

Task 11

Write a Java class Book with following features:

- Instance variables :
  - **title** for the title of book of type String.
  - **author** for the author's name of type String.
  - **price** for the book price of type double.
- Constructor:
  - **public Book (String title, String author, double price)**: A constructor with parameters, it creates the Author object by setting the the fields to the passed values.
- Instance methods:
  - **public void setTitle(String title)**: Used to set the title of book.
  - **public void setAuthor(String author)**: Used to set the name of author of book.
  - **public void setPrice(double price)**: Used to set the price of book.
  - **public String getTitle()**: This method returns the title of book.
  - **public String getAuthor()**: This method returns the author's name of book.
  - **public String toString()**: This method printed out book's details to the screen

Write a separate class **BookDemo** with a main() method creates a Book titled "Developing Java Software" with authors Russel Winderand price 79.75. Prints the Book's string representation to standard output (using System.out.println).

Task 12

Write a Java class Author with following features:

- Instance variables :
  - **firstName** for the author's first name of type String.
  - **lastName** for the author's last name of type String.
- Constructor:
  - **public Author (String firstName, String lastName)**: A constructor with parameters, it creates the Author object by setting the two fields to the passed values.
- Instance methods:
  - **public void setFirstName (String firstName)**: Used to set the first name of author.
  - **public void setLastName (String lastName)**: Used to set the last name of author.
  - **public String getFirstName()**: This method returns the first name of the author.
  - **public String getLastName()**: This method returns the last name of the author.
  - **public String toString()**: This method printed out author's name to the screen.

Task 13

Write a Java class Clock for dealing with the day time represented by hours, minutes, and seconds. Your class must have the following features:

- Three instance variables for the hours (range 0 - 23), minutes (range 0 - 59), and seconds (range 0 - 59).
- Three constructors:
  - default (with no parameters passed; is should initialize the represented time to 12:0:0)
  - a constructor with three parameters: hours, minutes, and seconds.
  - a constructor with one parameter: the value of time in seconds since midnight (it should be converted into the time value in hours, minutes, and seconds)
- Instance methods:
  - a *set*-method method **setClock()** with one parameter *seconds* since midnight (to be converted into the time value in hours, minutes, and seconds as above).
  - *get*-methods **getHours(), getMinutes(), getSeconds()** with no parameters that return the corresponding values.
  - *set*-methods **setHours(), setMinutes(), setSeconds()** with one parameter each that set up the corresponding instance variables.
  - method **tick()** with no parameters that increments the time stored in a Clock object by one second.
  - method **addClock()** accepting an object of type Clock as a parameter. The method should add the time represented by the parameter object to the time represented in the current object. The new time should be returned as a clock object.
  - Add an instance method **toString()** with no parameters to your class. toString() must return a String representation of the Clock object in the form "(hh:mm:ss)", for example "(03:02:34)".
  - Add an instance method **tickDown()** which decrements the time stored in a Clock object by one second.
  - Add an instance method **subtractClock()** that takes one Clock parameter and returns the difference between the time represented in the current Clock object and the one represented by the Clock parameter. Difference of time should be returned as a clock object.

Write a separate class **ClockDemo** with a main() method. The program should:

- instantiate a Clock object firstClock using one integer *seconds* since midnight obtained from the keyboard.
- tick the clock ten times by applying its *tick()* method and print out the time after each tick.
- Extend your code by appending to it instructions instantiating a Clock object secondClock by using three integers (hours, minutes, seconds) read from the keyboard.
- Then tick the clock ten times, printing the time after each tick.
- Add the secondClock time in firstClock by calling method addClock.
- Print both clock objects calling toString method

Create a reference thirdClock that should reference to object of difference of firstClock and secondClock by calling the method subtractClock().

Task 14

Write a Java class Complex for dealing with complex number. Your class must have the following features:

- Instance variables :
  - **realPart** for the real part of type double
  - **imaginaryPart** for imaginary part of type double.
- Constructor:
  - **public Complex ()**: A default constructor, it should initialize the number to 0, 0)
  - **public Complex (double realPart, double imaginaryPart)**: A constructor with parameters, it creates the complex object by setting the two fields to the passed values.
- Instance methods:
  - **public Complex add (Complex otherNumber)**: This method will find the sum of the current complex number and the passed complex number. The methods returns a new Complex number which is the sum of the two.
  - **public Complex subtract (Complex otherNumber)**: This method will find the difference of the current complex number and the passed complex number. The methods returns a new Complex number which is the difference of the two.
  - **public Complex multiply (Complex otherNumber)**: This method will find the product of the current complex number and the passed complex number. The methods returns a new Complex number which is the product of the two.
  - **public Complex divide (Complex otherNumber)**: This method will find the ... of the current complex number and the passed complex number. The methods returns a new Complex number which is the ... of the two.
  - **public void setRealPart (double realPart)**: Used to set the real part of this complex number.
  - **public void setImaginaryPart (double realPart)**: Used to set the imaginary part of this complex number.
  - **public double getRealPart()**: This method returns the real part of the complex number
  - **public double getImaginaryPart()**: This method returns the imaginary part of the complex number
  - **public String toString()**: This method allows the complex number to be easily printed out to the screen

Write a separate class **ComplexDemo** with a main() method and test the Complex class methods.

Task 15

Write a java class called **BoroInt** that uses a String value to store big integers and can carry out basic arithmetic operations on them.

Instance variable:

String val

Overloaded constructors:

1. default constructor: sets the value of val to "0"
2. Takes a string value, checks whether it contains any non numerical values, if yes, then throws **BoroIntErModdheNumberCharaArKisuDeyaJaiNaException** (which is off course a custom exception which you must create :P) … otherwise, copy the contents to val. ***Note:*** See method list below to see what you can use :P
3. Takes an integer value, converts it to string and stores it in val.
4. Takes a **BoroInt** object and copies the val of the parameter into it's own val

Methods:

public String trim(String _val)

Removes all spaces in the String and returns a String value without any spaces.

public boolean validValue(**String _val**)

Returns true if the String _val doesn't contain any non numerical characters and false otherwise.

public BoroInt add(BoroInt _val)

Adds the value in the instance variable of the parameter to it's own value and returns a new BoroInt object whose instance variable contains the result of the addition.

public BoroInt subtract(BoroInt _val)

Subtracts the value in the instance variable of the parameter from it's own value and returns a new BoroInt object whose instance variable contains the result of the subtraction.

public BoroInt multiply(BoroInt _val)

Multiplies the value in the instance variable of the parameter with it's own value and returns a new BoroInt object whose instance variable contains the result of the subtraction.

**** public BoroInt divide(BoroInt _val)**

Divides the value in it's own instance variable by the value in the instance variable of the parameter and returns a new BoroInt object whose instance variable contains the result of the subtraction.