Université d'Ottawa
Faculté de génie

École de science informatique
et de génie électrique

uOttawa

L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

School of Electrical Engineering
and Computer Science

Introduction to Computing II (ITI 1121)
Midterm Examination - SOLUTION

Instructor: Lucia Moura

June 2015, duration: 2 hours

Identification

Student name: _____

Student number: _____ Signature: _____

Instructions

1. This is a closed book examination;
2. No calculators, electronic devices or other aids are permitted;
   (a) Any electronic device or tool must be shut off, stored and out of reach;
   (b) Anyone who fails to comply with these regulations may be charged with academic fraud.
3. Write comments and assumptions to get partial marks;
4. Write your answers in the space provided.
   (a) Use the back of pages if necessary;
   (b) You may not hand in additional pages.
5. Do not remove the staple holding the examination pages together;
6. Beware, poor hand writing can affect grades;

Marking scheme

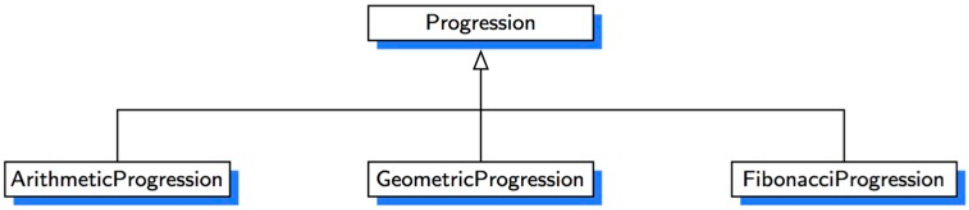
Question	Maximum	Result
1	25	
2	13	
3	12	
Total	**50**	

Question 1 (25 marks)

You are being considered for a job by a high school board to implement a numeric progression generator to be used by the students in their Math and Computing classes. At your job interview you are asked to implement the prototype described here. Your prototype should provide some basic numeric progressions and have a framework setup for others to create their own numeric progressions.

A numeric **progression** is a sequence of numbers where each number depends on one or more of the previous numbers. For example, an **arithmetic progression** determines the next number by adding a fixed constant to the previous value, and a **geometric progression** determines the next number by multiplying the previous value by a fixed constant.
(See examples at the bottom of the next page.)

The following class diagram gives you an overview of this application. Follow the instructions below.



A. (4 marks) Complete the implementation of the class **Progression** by implementing method **void printProgression(int n)** and adding some qualifying words to the class declaration.
Progression provides the characteristics that are common to its sub-classes. It declares three methods:

- A progression has a current value stored in the variable **current**.
- It has an abstract method **void advance()** which updates **current** to the next value; this is specific to the type of progression.
- It implements a method **nextValue()** which has been implemented for you.
  nextValue() prints the current value and advances the sequence to the next current value.
- You must add implementation of method **void printProgression(int n)** which **you must write**.
  printProgression(int n) prints the **n** next values in the sequence, also advancing the sequence.
- You must add appropriate words before the class declaration.

B. (6 marks) Implement the class **ArithmeticProgression**.
ArithmeticProgression is a specialized **Progression**.

- It contains a variable **long increment** which stores the increment of the arithmetic progression.
- It contains a constructor **ArithmeticProgression(long start, long increment)**.
  This constructor sets the current value to **start** and stores the **increment**.
- Method **void advance()** must update the current value in the sequence. The arithmetic progression obtains the next element by adding the increment to the current value.

C. (6 marks) Implement the class **GeometricProgression**.
GeometricProgression is a specialized **Progression**.

- It contains a variable **long base** which stores the number that multiplies the previous one in the geometric progression.
- It contains a constructor **GeometricProgression(long start, long base)**.
  This constructor sets the current value to **start** and stores **base**.
- Method **void advance()** must update the current value in the sequence. The geometric progression obtains the next element by multiplying the the current value by the base.

D. (9 marks) Implement the class **FibonacciProgression**.
FibonacciProgression is a specialized **Progression**.

- It has variable **long previous** which stores the previous element in the sequence, as this is necessary in the calculation of the next element.
- It contains a constructor **FibonacciProgression()**.
  This constructor initializes any necessary values. The current value must become the first in the Fibonacci sequence (which is 0).
- Method **void advance()** must update the current value in the sequence according to the Fibonnacci rule: The first two values in the Fibonacci progression are 0 and 1 and the following values are obtained by adding the two previous values (the current one and the one before): $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, \ldots$

Below you can see a class that tests the classes you will implement

```
public class ProgressionTest {

        public static void main(String[] args) {
                Progression[] a = new Progression[5];
                String [] name = new String[5];

                // creates five numeric progressions
                a[0]=new ArithmeticProgression(0,2);     name[0]="A(0,+2)  ";
                a[1]=new ArithmeticProgression(3,4);     name[1]="A(3,+4)  ";
                a[2]=new GeometricProgression(6,2);      name[2]="G(6,*2)  ";
                a[3]=new GeometricProgression(1,5);      name[3]="G(1,*5)  ";
                a[4]=new FibonacciProgression();         name[4]="Fibonacci";

                // print the first 10 elements of each numeric progression
                for (int i=0; i<5; i++) {
                    System.out.print(name[i]+": ");
                    a[i].printProgression(10);
                }

        }

}
```

Here is the expected output.

```
A(0,+2)  : 0 2 4 6 8 10 12 14 16 18
A(3,+4)  : 3 7 11 15 19 23 27 31 35 39
G(6,*2)  : 6 12 24 48 96 192 384 768 1536 3072
G(1,*5)  : 1 5 25 125 625 3125 15625 78125 390625 1953125
Fibonacci: 0 1 1 2 3 5 8 13 21 34
```

Progression

```
public abstract class Progression {

        protected long current;

        protected abstract void advance() ;

        public long nextValue() {
                long answer= current;
                advance();
                return answer;
        }

        // prints the next n values of the progression
        public void printProgression(int n) {
                System.out.print(nextValue());
                for (int i=1; i < n ; i++)
                        System.out.print(" "+nextValue());
                System.out.println();

        }



}
```

ArithmeticProgression

```
public class ArithmeticProgression extends Progression{

        protected long increment;

        public ArithmeticProgression(long start, long increment) {
         current = start;
         this.increment = increment;
        }

        protected void advance () {
                current = current + increment;
        }

}
```

GeometricProgression

```
public class GeometricProgression extends Progression {

        protected long base;

        public GeometricProgression(long start, long base) {
                current = start;
                this.base = base;
        }

        protected void advance () {
                current = current*base;
        }

}
```

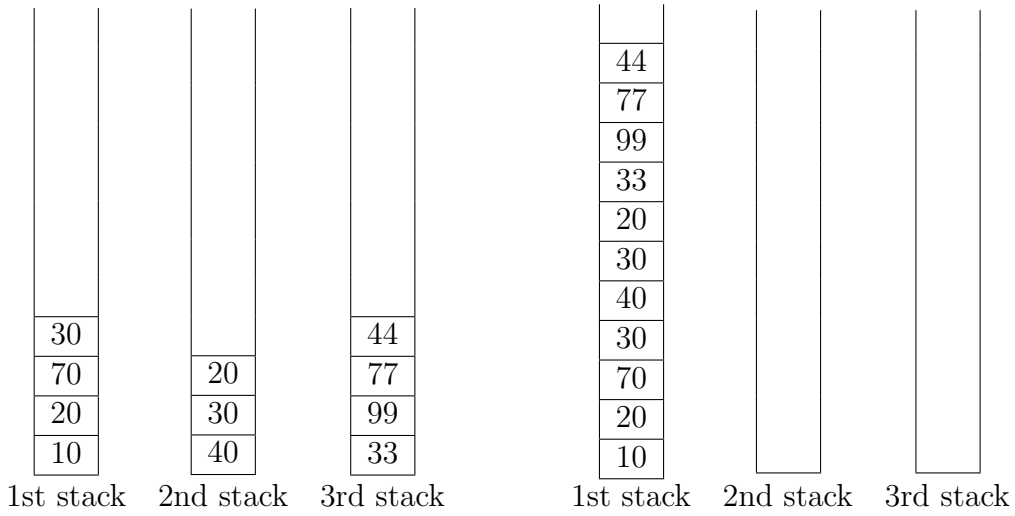
FibonacciProgression

```
public class FibonacciProgression extends Progression {

        private long prev;

        public FibonacciProgression() {
                current = 0;
                prev = 1;
        }

        protected void advance() {
                long temp = current;
                current = current+ prev;
                prev = temp;
        }

}
```

Question 2 (13 marks)

For the class **Q2**, you must implement the class method **void merge(Stack s1, Stack s2, Stack s3)**.

- The method **merge** combines the contents of the stack designated by **s1** with the contents of the stack designated by **s2** and with the contents of the stack designated by **s3** in the specific order explained next. The combined stack will have the original elements of each stack in their original order, placing the original elements of the first stack at the bottom, the ones of the second stack in the middle and the ones of the third stack at the top.



- After the call to the method **merge**, the stack designated by **s1** must contain the combined contents as described above, while the stack designated by **s2** and the stack designated by **s3** must be both empty.
- You can assume that **s1**, **s2** and **s3** will NOT be **null**.
- One or more stacks may be initially empty; this is not a problem.
- The parameters of the method **merge** are of type **Stack**, which is an interface.

For this question, there is an interface named **Stack**:

```
public interface Stack {
    public abstract void push(int item);
    public abstract int pop();
    public abstract boolean isEmpty();
}
```

• Note: the parameter of the method **push** and the return value of the method **pop** are of type **int**.
• You can assume that the actual parameters of method **merge** are each an instance of some class that implements Stack which can store an arbitrary large number of elements.
• You cannot use arrays to store temporary data. However, during the computations done inside method **merge** you may move elements between the stacks, as you please, by using the methods given in the interface **Stack**.
• You may declare variables of any primitive data type, but no new objects of other classes.
Write your answer on the next page.

```
public class Q2 {
    public static void merge(Stack s1, Stack s2, Stack s3) {


                int count2=0;
                while (!s2.isEmpty())  {
                        s3.push(s2.pop());
                        count2++;
                        }

                for (int i=0; i<count2; i++)
                        s1.push(s3.pop());

                while (!s3.isEmpty())
                        s2.push(s3.pop());

                while (!s2.isEmpty())
                        s1.push(s2.pop());


        }

    } // End of merge
} // End of Q2
```

Question 3 (12 marks) Short Answers

A. (5 marks) Multiple choice questions

Consider the following declarations (note that the classes are not complete)

```
package pack1;
public class Class1 {
     private int v1;
     protected int v2;
     int v3;
     public int v4;
}

package pack1;
public class Class2 { /* ... */ }

package pack2;
public class Class3  extends pack1.Class1{ /* ... */ }

package pack2;
public class Class4 { /* ... */ }
```

(a) Which variables of pack1.Class1 are visible in pack1.Class2 ?

i. v1, v2, v3, v4
ii. **v2, v3, v4, only**
iii. v2, v4, only
iv. v3, v4, only
v. none of the above

(b) Which variables of pack1.Class1 are visible in pack2.Class3 ?

i. v1, v2, v3, v4
ii. v2, v3, v4, only
iii. **v2, v4, only**
iv. v3, v4, only
v. none of the above

(c) Which variables of pack1.Class1 are visible in pack2.Class4 ?

i. v1, v2, v3, v4
ii. v2, v3, v4, only
iii. v2, v4, only
iv. v3, v4, only
v. **none of the above** (note: answer must be v4 only)

(d) _____ allows us to create new classes based on existing classes.

i. polymorphism
ii. **inheritance**
iii. method overloading
iv. the copy constructor
v. none of the above

(e) Consider the Java program below.

```
public class Test {
    public static void main(String[] args) {
        int[] a, b;
        a = new int [100];
        b = new int [100];

        for (int i=0; i < 100; i++) {
                a[i]=i;
                b[i]=i;
        }

        if (a==b)    System.out.print("1A, ");
        else         System.out.print("1B, ");

        b = a;
        b[0] = 100;

        if (a==b)  System.out.println("2A");
        else       System.out.println("2B");
    }

}
```

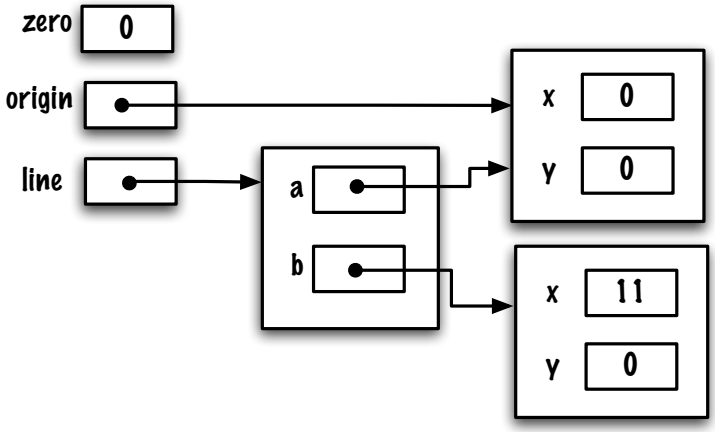
The program displays

i. 1A, 2A
ii. 1A, 2B
iii. **1B, 2A**
iv. 1B, 2B
v. None of the above because the program causes a runtime error.

B. (4 marks) Following the guidelines presented in class, as well as the lecture notes, draw the memory diagrams for all the objects and all the local variables of the method **Q3.test** following the execution of the statement "**line = new Line(origin, new Point(11, 21)**)".

```
public class Point {
    private int x = 0;
    private int y = 0;
    public Point( int x, int y ) {
        this.x = x;
        y = y;
    }
}

public class Line {
    private Point a;
    private Point b;
    public Line( Point a, Point b ) {
        this.a = a;
        this.b = b;
    }
}

public class Q3 {
    public static void test() {
        int zero;
        Point origin;
        Line line;
        zero = 0;
        origin = new Point( zero, 0 );
        line = new Line( origin, new Point( 11, 21 ) );
        // Here!
    }
}
```

Answer:



C. (3 marks) Short answers

(a) Give the result that will be printed on the standard output when the following **main** method is executed.

```
public class Ticket {
    private static int nextSerialNumber = 100;
    private int serialNumber;
    public Ticket() {
        serialNumber = nextSerialNumber;
        nextSerialNumber = nextSerialNumber + 1;
    }
    public int getSerialNumber() {
        return serialNumber;
    }
    public static void main( String[] args ) {
        Ticket t1, t2, t3;
        t1 = new Ticket();
        t2 = new Ticket();
        t3 = new Ticket();
        System.out.print( t1.getSerialNumber() + "," );
        System.out.print( t2.getSerialNumber() + "," );
        System.out.println( t3.getSerialNumber() );
    }
}
```

```
Answer: 100,101,102
```

(b) Give the result that will be printed on the standard output when the following **main** method is executed.

```
public class Ticket {
    private int nextSerialNumber = 100;
    private int serialNumber;
    public Ticket() {
        serialNumber = nextSerialNumber;
        nextSerialNumber = nextSerialNumber + 1;
    }
    public int getSerialNumber() {
        return serialNumber;
    }
    public static void main( String[] args ) {
        Ticket t1, t2, t3;
        t1 = new Ticket();
        t2 = new Ticket();
        t3 = new Ticket();
        System.out.print( t1.getSerialNumber() + "," );
        System.out.print( t2.getSerialNumber() + "," );
        System.out.println( t3.getSerialNumber() );
    }
}
```

```
Answer:  100,100,100
```

(c) Give the result that will be printed on the standard output when the following **main** method is executed.

```
public class Ticket {
    private static int nextSerialNumber = 100;
    private static int serialNumber;
    public Ticket() {
        serialNumber = nextSerialNumber;
        nextSerialNumber = nextSerialNumber + 1;
    }
    public int getSerialNumber() {
        return serialNumber;
    }
    public static void main( String[] args ) {
        Ticket t1, t2, t3;
        t1 = new Ticket();
        t2 = new Ticket();
        t3 = new Ticket();
        System.out.print( t1.getSerialNumber() + "," );
        System.out.print( t2.getSerialNumber() + "," );
        System.out.println( t3.getSerialNumber() );
    }
}
```

Answer: 102,102,102

(blank space)

(blank space)