

**CS101**  
**Algorithms and Programming 1**  
**Section 2**  
**Spring 2007**

**QUIZZES**  
**In Reverse Chronological Order**

**(Shorter answers are accepted, here answers are  
given in more detail than required.)**

**CS101: Algorithms and Programming I****Quiz No. 11 (Take Home)**

May 2, 2007

Due Date: May 16, 2007, by 5:00 pm (Leave it in my mailbox in the department)

**Notes:** For submission please follow the guidelines provided for the previous take home quizzes.

**A. Purpose**

More experience with object-oriented programming and introduction to arrays.

**B. Description**

Implement the `intVector` class as defined below and test all of its methods : (implementation of some of the methods are given). You have to fix the errors (if any) of the given methods. The standard deviation

formula is given in the following; in this formula  $n$  indicates the number of elements and  $\bar{x}$  indicates the average value. For  $n = 1$  the standard deviation is 0 by definition.

$$\text{standard deviation} = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{(n-1)}}$$

```

/**
 * Write a description of class intVector here.
 *
 * @author Your Name
 */
public class intVector
{
    int[ ] array;
    static final int ERROR_CODE= 999;
    // It is assumed that ERROR_CODE will not be used as an element value
    // (no need to verify this during construction insertion, etc).

    public intVector( )
    {
        array= null;
    }
    //=====
    public intVector(int[ ] inArray)
    {
        array= new int[inArray.length];
        for(int i= 0; i < inArray.length; i++)
            array[i]= inArray[i];
    }
    //=====
    // Add an element to the end of this intVector
    public void addElement(int ele)
    {
        int[ ] temp= new int[array.length + 1];
        for (int i= 0; i < array.length; i++)

```

```

        temp[i]= array[i];
        temp[temp.length - 1]= ele;
        array= temp;
    }
//=====
// Add otherVector and this intVector, if needed increase the size and
// return an intVector as the result of the operation.
// {1, 2} + {2, 3, 4} produces {3, 5, 4}.
    public intVector add(intVector otherVec)
    {
        int size1= array.length;
        int size2= otherVec.array.length;
        // add a number to the end of this intVector
        int size= size1 > size2 ? size1 : size2;

        int temp[ ]= new int [size];

        if(size1 > size2) {
            for(int i= 0; i < size2; i++)
                temp[i]= array[i] + otherVec.array[i];
            for(int i= size2; i < size1; i++)
                temp[i]= array[i];
        }
        else {
            // Complete this part.
        }

        intVector r= new intVector(temp);
        return(r);
    }
//=====

// Assigns a copy of otherVec to this intVector
    public void assign(intVector otherVec)
    {
        if(otherVec.array == null)
            this.array= null;
        else{
            array= new int [otherVec.array.length];
            for(int i= 0; i < array.length; i++)
                array[i]= otherVec.array[i];
        }
    }
//=====
// This assignment would cause some problems.  See the tester program.
    public void assign2(intVector otherVec)
    {
        array= otherVec.array;
    }
//=====
// Finds the average value of intVector elements.
// If it is empty it returns ERROR_CODE as average value.
    public double average( )
//=====
// Returns a true value if this intVector contains arg.
    public boolean contains(int arg)

```

```

=====
// Returns an identical copy of this intVector.
public intVector copy()

=====
// Returns the element at index position, if there is no index like that
// it returns ERROR_CODE value as its result.
public int elementAt(int index)

=====
// Returns true if contents of this intVector is the same as the contents of
// otherVector. Note that the order of the elements is insignificant: {1, 2, 3}
// and {2, 3, 1} are assumed to be equal by this method.
public boolean equals(intVector otherVector)

=====
// Returns the value of the first element in this intVector.
// Returns ERROR_CODE if no such element (i.e., array is empty).
public int firstElement()

=====
// Returns the index of the first occurrence of element arg.
// A -1 is returned if the element is not found.
public int indexOf(int arg)
{
    int position= -1;
    if(array != null)
        for(int i= 0; i < array.length && position == -1; i++)
            if(array[i] == arg) position= i;
    return(position);
}

=====
// Returns the index of the first occurrence of element arg starting at
// index. A -1 is returned if the element is not found.
public int indexOf(int arg, int index)

=====
public boolean insertElementAt(int arg, int index)
// Adds element arg at the specified index, increments intVector length,
// and returns true. If the specified index does not exist returns false.
{
    boolean r= array.length > 0 && index >= 0 && index <array.length;
    if(r) {
        int[] temp= new int[array.length + 1];
        for(int i= 0; i < index; i++)
            temp[i]= array[i];
        temp[index]= arg;
        for(int i= index; i < array.length; i++)
            temp[i+1]= array[i];
        array= temp;
    }
    return(r);
}

=====
// Returns true if arg is an element of this intVector
public boolean isElement(int arg)
=====
// Returns true if array contains no elements.
public boolean isEmpty()
{

```

```

    return(array == null);
}
//=====
// Returns the value of the last element in this intVector.
// Returns ERROR_CODE if no such element (i.e., array is empty).
public int lastElement()
{
    int lastValue= (isEmpty()? ERROR_CODE : array[array.length-1]);
    return(lastValue);
}
//=====
// Returns the first index that arg occurs at this intVector, starting
// a backwards search. A -1 is returned if arg is not found.
public int lastIndexOf(int arg)
//=====
// Returns the first index that arg occurs at this intVector, starting
// a backwards search at the specified index.
// A -1 is returned if arg is not found.
public int lastIndexOf(int arg, int index)
{
    int position= -1;
    if(array != null && index < array.length)
        for(int i= index; i >= 0 && position == -1; i--)
            if(array[i] == arg) position= i;
    return(position);
}
//=====
// Returns the value of the maximum element of this intVector.
// If it is empty it returns ERROR_CODE as maximum value.
public int max()
//=====
// Returns the value of the minimum element of this intVector.
// If it is empty it returns ERROR_CODE as minimum value.
public int min()
//=====
// Removes all elements.
public void removeAllElements()
{
    array= null;
    // Garbage collection is done automatically.
}
//=====
// Removes all occurrences of arg from this intVector and returns a true value.
// The size of this intVector shrinks after removal.
// If there is no occurrence of arg it returns a false value.
public boolean removeElement(int arg)
//=====
// Reverses the contents of this intVector,
// for example { 1, 2, 3 } becomes { 3, 2, 1 }
public void reverse()
//=====
// Sets the element at the specified index equal to arg and returns true,
// if index position does not exist keeps this intVector the same
// and returns false.
public boolean setElementAt(int index, int arg)
{

```

```

    boolean r= index >= 0 && index < array.length;
    if(r)
        array[index]= arg;
    return(r);
}
=====
// Returns the size of this intVector
public int size( )
{
    return( ((array == null)? 0 : array.length) ); // try a simpler implementation.
}
=====
// Sorts the elements of this intVector in ascending order using the bubble sort algorithm.
public void sort( )
{
    // An empty array is sorted by definition.
    if(array.length > 0) {
        // Bubble sort: Compare ith element i+1st element if
        // ith > i+1st interchange them and do this as much as
        // needed.
        int ind;
        int pass= 1;
        do {
            ind= 0;
            for(int i= 0; i < array.length - pass; i++)
                if(array[i] > array[i+1]) { // if so swap (exchange)
                    int temp= array[i];
                    array[i]= array[i+1];
                    array[i+1]= temp;
                    ind= 1; // there was a swap
                }
            pass++;
        } while(ind == 1 && pass <= array.length);
    }
}
=====
// Check if this intVector elements are sorted.
// Both ways of sorting (ascending or descending) are acceptable.
public boolean sorted( )
=====
// Returns true if this intVector is sorted in ascending order:
// array[i] <= array[i+1] for (0 <= i < (array.length - 1)).
// By definition an empty intVector is sorted.
public boolean sortedAscending( )
=====
// Returns true if this intVector is sorted in descending order:
// array[i] >= array[i+1] for (0 >= i < (array.length - 1)).
// By definition an empty intVector is sorted.
public boolean sortedDescending( )
=====
// Returns standard deviation of the elements of this intVector.
// Returns ERROR_CODE as the result if this intVector is empty.
public double std( )
=====
// Subtract otherVector from this intVector, if needed increase the size and
// returns an intVector as the result of the operation.

```

```

// {5, 6} - {1, 2, 3} produces {4, 4, -3}.
// {1, 2, 3} - {5, 6} produces {-4, -4, 3}.
    public intVector sub(intVector otherVec)
//=====
// Returns a string representation of this intVector.
    public String toString()
    {
        String S= "";
        for(int i= 0; i < array.length; i++)
            S= S + "\n" + array[i];
        return(S);
    }
//=====
}

```

### C. When and How to Submit Your Work

Note that hard coded main( ) method is acceptable (i.e., if there is no user interaction it is OK); however, you have to make sure that each method of both classes are tested at least once. An **incomplete** tester class is given in the following box, please modify and use it as needed.

```

/**

* Write a description of class testIntVector here.

*

* @author Fazli Can

* @version (a version number or a date)

*/

import java.text.DecimalFormat;

```

```
public class testIntVector
{

    public static void main (String [ ] args)
    {

        DecimalFormat dfmt = new DecimalFormat("0.###");

        int[ ] arr1= {3, 2, 5, 3};

        int[ ] arr2= {1, 2, 3};

        intVector a= new intVector(arr1);

        intVector b= new intVector(arr2);

        a.sort( );

        a.reverse( );

        System.out.println("a=====>" + a);

        a.addElement(7);

        a.setElementAt(1, 50);

        System.out.println("a=====>" + a);

        System.out.println("b=====>" + b);

        System.out.println("a.add(b)=====>" + a.add(b));
```

```
System.out.println("a.std( ):" + dfmt.format(a.std( )));
```

```
intVector c= new intVector(arr2);
```

```
intVector d= new intVector(arr1);
```

```
d.assign2(c);
```

```
d.setElementAt(1, 500);
```

```
System.out.println("c =====>" + c);
```

```
System.out.println("d =====>" + d);
```

```
// Changing d also changes c: the assign2( ) method is no good. WHY?
```

```
intVector x= new intVector(arr2);
```

```
intVector y= new intVector(arr1);
```

```
y= x;
```

```
y.setElementAt(1, 500);
```

```
System.out.println("x =====>" + x);
```

```
System.out.println("y =====>" + y);
```

```
// Changing y also changes x: the = operator is no good. WHY?
```

```
intVector alfa= new intVector(arr2);
```

```
intVector beta= new intVector(arr1);

alfa.assign(beta);

alfa.setElementAt(1, 500);

System.out.println("alfa =====>" + alfa);

System.out.println("beta =====>" + beta);

// Changing alfa does not affect beta: the assign( ) method is good. WHY?

}

}
```

Output of the above program is given in the following box.

```
a=====>
```

```
5
```

```
3
```

```
3
```

```
2
```

```
a=====>
```

```
5
```

```
50
```

3

2

7

b=====>

1

2

3

a.add(b)=====>

6

52

6

2

7

a.std():41,1

c=====>

1

500

3

d =====>

1

500

3

x =====>

1

500

3

y =====>

1

500

3

alfa =====>

3

500

5

3

beta =====>

3

2

5

3

**Quiz No. 10**  
**April 25, 2007**

Write copy() method for the class RationalNumber (see Lewis & Loftus, 5<sup>th</sup> ed., pp. 332-335 for the class definition).

```
public RationalNumber copy ( )
{
    RationalNumber r;
    r= new RationalNumber( numerator, denominator);
    return(r);
}
```

A shorter implementation:

```
public RationalNumber copy ( )
{
    return( new RationalNumber( numerator, denominator) );
}
```

**Quiz No. 9**  
**April 20, 2007**

Write a method to find the following summation.

$$sum = 1 + \frac{1}{2!} + \frac{1}{3!} + \dots + \frac{1}{n!}$$

Where  $n$  is of type `int` and the input parameter. In the method continue adding new elements as long as the added item is  $\geq 0.001$ .

Solution: provides a nested loop example.

```
public float sum(int n) // n >= 1
{
    float s, factor, element;
    int i;

    s= 0.0;
    element= 1.0;
    i= 1;

    do
    {
        s= s + element;

        // Calculate the next element:
        i++;
        factor= 1.0;
        for(int j= 1; j <= i; j++)
            factor= factor * j;
        element= 1.0 / factor;

    } while(i <= n && element >= 0.001);

    return(s);
}
```

The efficiency of the above implementation can be improved!

Hint: Do we need to calculate  $i!$  from the very beginning?

**Quiz No. 8 (this is a take-home Quiz)****April 12, 2007****Due date: April 20, 2007, Friday, Class Time**

Please implement the following three programs. You will submit hardcopy of your work (i.e., a paper submission is required). For your submission, please follow the guidelines provided below. Make sure that your programs contain proper comments and are easy to understand. Do not forget to intend your code properly. In the programs, make sure that your name is shown as the author. State your assumptions (if any) in the comments.

**Requirement:** In your implementations you have to use the following loop statements at least once: for, while, and do-while.

**Guidelines for paper submission**

1. Print your program listings,
2. Print the execution time screen shots for your programs (note that you can get a copy of the screen to the clipboard by pressing the keys Alt-PrntScn at the same time),
3. Prepare a table of contents page and put your name and provide a list of contents with page numbers,
4. Label all printouts and screen shots properly for efficient grading,
5. Note that there are three programs to write,
6. Staple all papers and submit your work as one piece.

**Program No. 1: Taylor Series Approximation of Functions**

Calculate the approximate values of the  $\sin()$ ,  $\cosine()$ , and  $\ln()$  functions at a given point (i.e., the  $x$  value) by using their Taylor series expansion given below.

a)

$$\sin x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots \text{ for all } x$$

b)

$$\cos x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \dots \text{ for all } x$$

c)

$$\ln(1+x) = \sum_{n=0}^{\infty} \frac{(-1)^n}{n+1} x^{n+1} \quad \text{for } |x| < 1$$

Continue adding the terms (each added item is referred to as “term”) to the summation variable and stop when the difference between two consecutive added terms is less than or equal to  $10^{-3}$ .

When you run your program, the user should be able to select the function, and after choosing the function name user will enter a value.

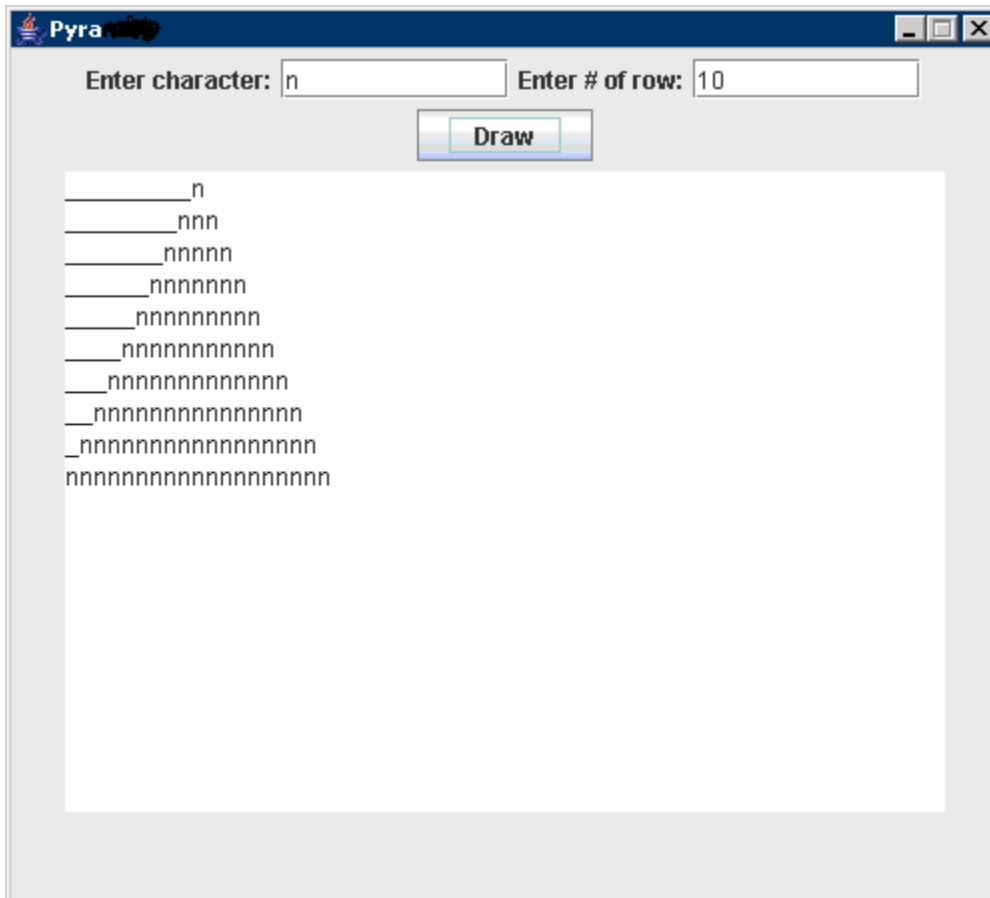
Your program must output the Taylor series approximation and the value calculated by the Math class methods. Make sure that your program is user friendly. The implementation can be GUI-based or console-based application.

### Program No. 2: Drawing a Pyramid

Write a program that draws a pyramid according to the following specifications:

- the height(# of rows) of the pyramid,
- the character to be used for drawing.

The program output should be as follows.



**Program No. 3: Robot**

Define a class “Robot” as explained below.

Two integer coordinates give the location of a robot: x for its east-west position and y for its north-south position (x increases as the robot moves east and y increases as the robot moves north).

The robots can face the following four directions: east, west, north, south.

The direction concept should be implemented using an enumerated user defined data type called Direction.

The public interface for a robot is as follows (i.e., these are all public member methods):

```
public Robot( int x, int y, Direction d);
public Robot( ); // default constructor sets the initial coordinates as 0, 0, and the direction as
“east”
public void move( int distance );
public void turnLeft( );
public void turnRight( );
public int xPosition( );
public int yPosition( );
public Direction orientation( );
public float distance(Robot otherRobot);
public Robot copy( );
public String toString( );
```

The constructor creates a robot in a given location and facing a given direction.

The method move( ) moves the robot a given distance in the direction it is facing; this is done by adding or subtracting the distance from the appropriate coordinate.

The methods turnLeft( ) and turnRight( ) turn the robot to the left or right by 90 degrees.

xPosition( ), yPosition( ), and orientation( ) return the robot's current location and the direction in which it is facing.

distance( ) returns the distance between this and the other robot.

In your implementation, make sure that you use the case statement. Hint: turnLeft( ) and turnRight( ) are two good opportunities for this purpose.

Note that this program requires no animation or GUI. If you like you are allowed to use GUI or animation.

Implement an interactive main( ) method to test your robot. First, get the initial conditions, after that perform the following. (You may assume that user input is always correct.)

1. Ask the user if he/she wants to turn the robot. If the user response is "Yes" ask where to turn. The user response can be “Left” or “Right”. Note that these user responses must be stored in String variables.
2. Ask the user to enter the distance to be traveled. The user response can be any integer  $\geq 0$ .
3. Display current coordinates and direction using orientation. Use toString( ) for this purpose.

4. If the last "distance to be traveled" is greater than zero go to step 1, otherwise stop moving this robot.
5. Create another robot and display the distance between two robots.
6. Copy one of the robots and show that `copy()` works fine.

**CS101/Section 2****Quiz No. 7 (this is a take-home Quiz)****March 29, 2007****Due date: April 4, 2007, Wednesday, Class Time**

Please implement the following three programs. You will submit hardcopy of your work (i.e., a paper submission is required). For your submission follow the guidelines provided below. Make sure that your programs contain proper comments and are easy to understand. Do not forget to intend your code properly. In the programs make sure that your name is shown as the author. Any assumptions should be stated in the comments.

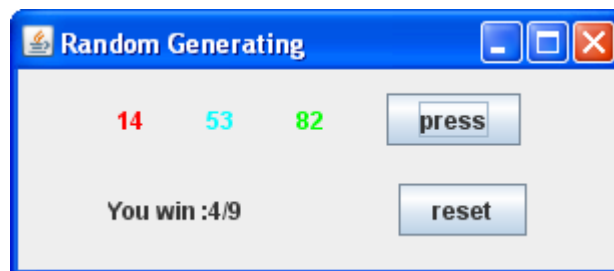
**Guidelines for paper submission**

1. Print your program listings,
2. Print the execution time screen shots for your programs (note that you can get a copy of the screen to the clipboard by pressing the keys Alt-PrntScn at the same time),
3. Prepare a table of contents page and put your name and provide a list of contents with page numbers,
4. Label all printouts and screen shots properly for efficient grading,
5. Note that there are three programs to write,
6. Staple all papers and submit your work as one piece.

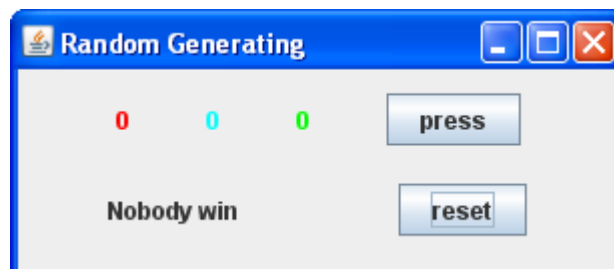
**Programs****Program No. 1: GUI-based Number Game**

Write a class that provide the following user interface and actions.

1. The user plays the game by clicking the “press” button. Each such click generates 3 random integer numbers between 1 and 100. These numbers are displayed in different labels with different colors.
  - a. If the summation of these three numbers exceeds 150 the user wins otherwise the computer wins.
  - b. After each game display the number of user wins over total number of games.



2. When the user clicks “reset” button the game resets itself as follows.



**Program No. 2: Circle and Rectangle**

Write the Circle and Rectangle classes as defined below.

The Circle class has a variable that keeps the length of the radius. The Rectangle class has width and height variables. In addition, both of these classes must have methods that calculate the area and circumference. You may need to have some additional class variables. If you like you can define additional class variables to perform the following operations.

**Circle**

- Constructor (radius must be the input parameter).
- get and set methods for radius.
- get methods for area and circumference.
- `isEqual(Circle)` method to check whether two circle is equal or not. This method takes Circle object as an input and returns true if this circle is equal to the input circle, a possible use is like this: `myCircle.isEqual(yourCircle)`.
- `isBigEnoughToHold(Rectangle)` method to check whether the circle is big enough to hold the input rectangle in or not.
- `toString()` method which prints the specification of the circle objects.

**Rectangle**

- constructor (width and height must be the input parameters).
- get and set methods for width and height.
- get methods for area and circumference.
- `isEqual(Rectangle)` method to check whether two rectangle is equal or not. This method takes another Rectangle object as an input and returns true if the rectangle is equal to the input rectangle. Note for example that a rectangle with “width=3” and “height=5” will be equal to a rectangle with “height =5” and “width =3”.
- `isBigEnoughToHold(Circle)` method to check whether the rectangle is big enough to hold the input circle in or not.
- `toString()` methods which prints the specification of the rectangle objects.

Finally, write a driver class to use all these functions of these classes.

**Important!**

Note that this problem requires no graphical representation, i.e., you do not need to display the objects.

**Program No. 3: Number Game**

Think about a game that is played with two people. The rules of the game are as follows:

1. One person says a number between 0 and maxIncrement. Then the game starts.
2. The other person increments this number by a number between 0 and maxIncrement parameter (this parameter will be set before starting the game).
3. The game ends when one of the players comes or exceeds to a number called winNumber (like maxIncrement this parameter will also be set before starting the game).
4. The game starts with setting the parameters (maxIncrement,winNumber, player 1 name, player 2 name and anything else that can be necessary for the game)
5. Then, the game starts from random selection of one of these players.
6. When player wants to enter a number, its name must be written on command prompt that waits for increment, after entering increment the incremented number should be printed, and if the game doesn't finish the turn comes to other player. The first coming to winNumber will win the game and the program will finish execution with declaring the winner of the game.
7. Make sure that you control the input number taken from the users, if it exceeds the maxIncrement the system should warn the user to reenter the input again.
8. For example if we set maxIncrement to 6 and winNumber to 20 and names are X and Y then (note with random selection X starts game):

```
This is X's turn: 5
The number is now 5,
This is Y's turn: 4
The number is now 9
This is X's turn: 3
The number is now 12,
This is Y's turn: 8
Error you exceed the maximum number please enter again:5
The number is now 17,
This is X's turn: 6
The number is now 23,
You win X. Congratulations! :-)
```

**Quiz No. 6**  
**March 28, 2007**

**Complete the constructor and withdraw() method.**

```
public class CreditCardAccount
{
    // Limit of the credit card
    int limit = 0;

    // Current expenses which has been done until now (monthly).
    int currentExpenses = 0;

    // Name of the account owner
    String accountOwner;

    // Constructor which takes two parameter, name and income of
    // the account owner.
    // The constructor must set accountOwner variable.
    // Limit of the account will be 350 for the income value which is
    // less than 1000, and limit of the account will be two times of
    // income value if income is greater than 1000

    CreditCardAccount(String name, int income)
    {

    }

    // Withdraw method checks the expenses of the account and
    // if it is appropriate, withdraws the amount from the account.
    // Returns whether the operation is done successfully or not.
    public boolean withdraw (int amount)
    {
        boolean result;

        int expensesAfterWithdraw = currentExpenses + amount;

        if(_____ )
        {
            currentExpenses = expensesAfterWithdraw;
            _____;
        }
        else
        {
            result = false;
        }

        return (result);
    }
}
```

**Quiz No. 6 Solution**

```
CreditCardAccount(String name, int income)
{
    accountOwner= name;
    if(income < 1000)
        limit= 350;
    else
        limit= 2 * income;
}

// Withdraw method checks the expenses of the account and
// if it is appropriate, withdraws the amount from the account.
// Returns whether the operation is done successfully or not.
public boolean withdraw (int amount)
{
    boolean result;

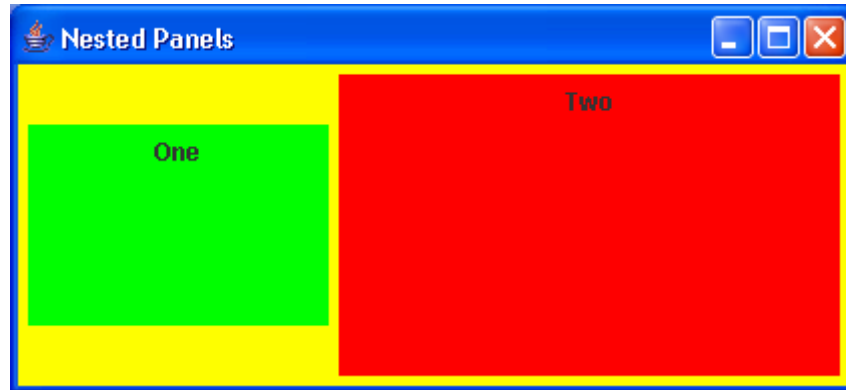
    int expensesAfterWithdraw = currentExpenses + amount;

    if(expensesAfterWithdraw <= limit)
    {
        currentExpenses = expensesAfterWithdraw;
        result= true;
    }
    else
    {
        result = false;
    }

    return (result);
}
}
```

**Quiz No. 5**  
**March 9, 2007**

Write a complete program to generate the following panel.



The frame name must be “Nested Panels”.

The primary panel inside this frame must be yellow and contain left and right panels.

The left panel must be green and have x and y dimensions as follows: 150, 100 pixels. It must contain a label that contains the word “One”.

The right panel must be red and have x and y dimensions as follows: 250, 150 pixels. It must contain a label that contains the word “Two”.

```
//*****
//  NestedPanels.java
//
//  Demonstrates a basic component hierarchy with nested panels.
//*****
import java.awt.Color;
import java.awt.Dimension;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class NestedPanel {
    public static void main(String args[])
    {
        JFrame mainFrame=new JFrame("Nested Panels");
        mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Set up first subpanel
        JPanel subPanel1 = new JPanel();
        subPanel1.setPreferredSize (new Dimension(150, 100));
```

```
subPanel1.setBackground (Color.green);
JLabel label1 = new JLabel ("One");
subPanel1.add (label1);

// Set up second subpanel
JPanel subPanel2 = new JPanel();
subPanel2.setPreferredSize (new Dimension(200, 150));
subPanel2.setBackground (Color.red);
JLabel label2 = new JLabel ("Two");
subPanel2.add (label2);

// Set up primary panel
JPanel primary = new JPanel();
primary.setBackground (Color.yellow);
primary.add (subPanel1);
primary.add (subPanel2);

mainFrame.getContentPane().add(primary);
mainFrame.pack();
mainFrame.setVisible(true);
}
}
```

**Quiz No. 4**  
**March 7, 2007**

**Questions**

1. Why doesn't the String class have to be specifically imported into our Java programs?
2. Given a Random object called rand, what value will be assigned to the variable i in the following statement?  
`int i= rand.nextInt(10);`
3. Write the declaration of an enumerated type that represents movie ratings.

**Answers** (shorter answers are acceptable)

1. The String class is not a part of the Java language. It is a part of the java.lang package. In order to use the classes of java.lang no import declaration is required.
2. [0, 9]
3. `enum movieRating {G, PG, PG13, R, NC17} //`  
[http://en.wikipedia.org/wiki/MPAA\\_film\\_rating\\_system](http://en.wikipedia.org/wiki/MPAA_film_rating_system)

Other reasonable answers are acceptable.

**Quiz No. 3**  
**March 2, 2007****Question (Problem)**

Write a complete program (class & main( )) to declare an object of type String that keeps your name and then

- Print your name and its length,
- Use substring( ) to assign your last name to another String object (String reference variable),
- Print your last name.

```
// Program for parsing a pre-defined String
// Written by Sarp Ulas Ocak
```

```
Public class nameParse
{
    public static void main (String[] args)
    {
        String fullName= new String("Sarp Ulas Ocak");

        int nameLength= fullName.length( );

        String lastName= new String(fullName.substring(10, 14));

        System.out.println("My name is " + fullName + "\n it is " +
            nameLength + " characters long \n and my last name is " + lastname);

    } // end of main( )
} // end of nameParse class
```

**Quiz No. 2**  
**February 7, 2007****Questions**

1. Define divide and conquer.
2. Define IDE.
3. Convert  $10011_2$  to decimal (equivalent number in base 10).
4. What is syntax (what does it mean in programming)?

**Answers**

1. Dividing a problem into smaller sub-problems and solving these sub-problems and integrating these sub-problem solutions to obtain the solution to the original problem is referred to as the “divide and conquer” approach (divide → obtain sub-problems, conquer → solve/defeat each sub-problem separately).
2. IDE (Integrated Development Environment) is a type of software used for program development. An IDEs provide all types of software (editor, compiler, loader, and debugger) needed for program development. Example: BlueJ, Eclipse, and JCreator.
3. Use the positional value of each bit location: from right to left the positional values are  $2^0=1$ ,  $2^1=2$ ,  $2^2=4$ ,  $2^3=8$ ,  $2^4=16$  →  $10011_2 = 16 + 2 + 1 = 19$ .
4. In programming syntax implies rules on coding. For example, in Java programming statements must end with a semicolon (;). If it is missing then it is a typo or a syntax error. For example, the Java statement “a= b + c, “ has a syntax error, since it does not end with a “;”. The syntax errors are flagged by the compiler during compilation time.

**Quiz No. 1**  
**January 31, 2007**

**Questions**

1. What are the major components of a computer?
2. What is paging?
3. What is multiprogramming?

**Answers**

1. CPU, Memory, I/O devices.
2. The operating system (OS, e.g., Windows, Unix) divides a program into equal parts called pages. During execution only the pages needed are kept in the main memory. If a new page is needed it is brought from disk. In this process if needed a page allocated to the program may be replaced.
3. Keeping more than one program active. For this purpose OS allocates the CPU time to each active program in turn.