Homework/Program #3 Solutions

1 What is the difference between an accessor method and a mutator method for a class?

(2.5 points for each correct definition below)

Accessor method returns some element of state of a particular Instance
For Example: FilledOval.getX()

Mutator method changes the state of a particular instance
For Example: FilledOval.moveTo()

2 Go to the book website

http://eventfuljava.cs.williams.edu/library/objectdrawJavadocV1.1.2/index.html

Looking at the documentation for the objectdraw library, How many different ways are there to construct a FramedRect object? What are the signatures of the constructor(s).
If you have a Text instance, what method is called to change the font size of the String to be displayed?

If you have VisibleImage instance, what method would you call to determine if the image should be displayed on the canvas?

(1 point for each correct bold statement below)

There are 3 different ways to construct FramedRect Object and following are the signatures:

FramedRect(double x, double y, double width, double height, DrawingCanvas canvas)

FramedRect(Location origin, double width, double height, DrawingCanvas canvas)

FramedRect(Location p0, Location p1, DrawingCanvas canvas)

setFontSize method is used to change the font size of the String to be displayed

Method signature of setFontSize is void setFontSize(int size)

isHidden() method is used to determine if the image should be displayed on the canvas

Method signature of isHidden() is boolean isHidden()

3. **Read section 5.5.2 of your book. It describes using Math.round() to convert doubles to integers**
(Java will not do this type conversion automatically).
Look at http://docs.oracle.com/javase/6/docs/api/java/lang/Math.html to see a full description of methods in the Math class. What do ceil() and floor() do? These methods return doubles, If you want an integer (long) version of these two functions, just using the Math class functions, how would you accomplish that? In other words, in the following code segment how would you compute low and a high so that they hold the floor and ceiling of x, respectively?

double x = 10.35;
long low, high;

(1 point each for def of floor,ceil, 3 points for correctly using round)

floor()- Returns the largest (closest to positive infinity) `double` value that is less than or equal to the argument and is equal to a mathematical integer.

ceil() - Returns the smallest (closest to negative infinity) double value that is greater than or equal to the argument and is equal to a mathematical integer.

round() - returns the closest `long` to the argument.

Just using Math class functions then to return long versions of these functions:
Math.round(Math.floor(x)) would evaluate to a long and give floor(x)
Math.round(Math.ceil(x)) – would evaluate to a long and give ceiling(x)

(Aside: Java can also do something called a typecast
We can explicitly typecast the value returned by floor and ceil to long , and Store it in low and high respectively

low = (long) Math.floor(x);
high = (long) Math.ceil(x);

4 Download Key.java, Lock.java, SampleKey.java from Lecture 7. Compile and run them. Now change the line in Key.java

myLock = theLock;

to

myLock = new Lock();
recompile Key.java then and re-run SampleKey. What does the unmodifed program print out?
What does the modified program print out. Explain why the output is different after the one line code change?

(2 point for correct output, 3 points for correct reason)

unmodified program prints "These are the same Object"
modified program doesnt print anything

There is a difference in the output because "new Lock()" will create a new instance of the Lock , so Lock returned by getMyLock() is different from the actual Lock
but theLock was using the Lock for the given key using "this" keyword(look in createKey() function), so Lock returned by getMyLock() is same as the actual Lock.

5 Gathering arguments from the command line. in the following, we'll use a very primitive way to gather arguments from the command line and make them usable variables in our programs. Later, we'll learn more compact ways to read arguments. Type in, compile and run the following program

```
public class ReadArgs
{
   static public void main(String [] args )
   {
      int numargs = args.length; // # arguments typed in
      int argindex = 0;
      int argvalue;
      while ( argindex < numargs )
      {
         argvalue = Integer.parseInt(args[argindex]);
         System.out.println("Argument " + argindex +":" + argvalue);
         argindex++;
      }
   }

}
```

To see what the program does, run with
java ReadArgs 20 200 50

and then
java ReadArgs 10 20

Note that Integer.parseInt() is defined by Java to convert a String into an integer.
There exists a similar set of options for the Double class (for example, http://docs.oracle.com/javase/6/docs/api/java/lang/Double.html, and http://docs.oracle.com/javase/6/docs/api/java/lang/Integer.html). Note: a Double object type is different than a double primitive type.
args[argindex] retrieves one element of the args array (We haven't done arrays yet).
args[0] is the first argument typed in on the command, line, args[1] is the second one.
Finally, args.length tells us how many arguments were typed in.

Modify the code to

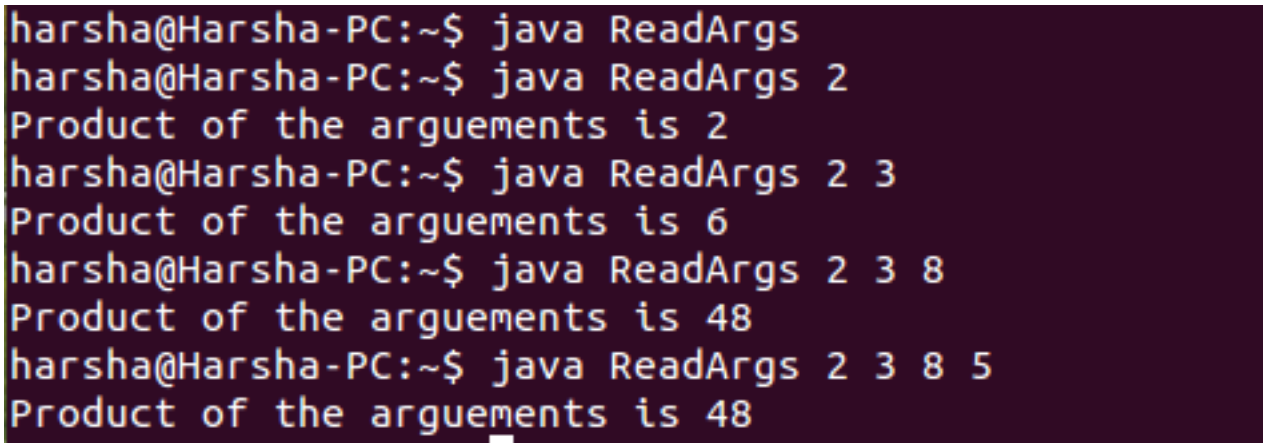
1. read no more than 3 arguments and call them arg1, arg2, arg3. These should each be of type int
2. instead of printing the arguments, print the product of the arguments (only calculate the the product of the first 3 arguments in the case where more than 3 were typed in at the command line). If less than three arguments are typed in, calculate the product of the arguments given to you.
3. should print nothing if no arguments are listed

Include code
Include output of your modified program when run with 0,1,2,3, and 4 arguments given on the command line

5 points for correct code. 1 point for each correct output

class ReadArgs{

```
static public void main(String[] args)
{
int numargs;
int argindex=0;
int argvalue;
int argprod=1;

if (args.length<3)
	numargs=args.length;
else
	numargs=3;
while (argindex<numargs)
	{
		argvalue=Integer.parseInt(args[argindex]);
		argprod=argprod*argvalue;
		argindex++;
	}
if (args.length>0)
	System.out.println("Product of the arguements is "+argprod);
}
}
```



6

T F All variables are initialized to zero by java
T F void sign(double x); and void sign(double y); have identical signatures
T F Constants must be declared public
T F A static variable is only visible with the class that defines it
T F It is legal use the private access modifier to declare a temporary variable within the statement block that defines a method
T F If a java source file defines a public class called BigClass, the file must be called BigClass.java
T F It is legal to perform an assignment within a boolean expression
T F Variables of type int are automatically converted to doubles, when needed.

T F Strings are a primitive type.
T F A class can only define one constructor

1 point for each correct response + 1 point (#4 should NOT be graded)
All variables are initialized to zero by java- False
void sign(double x); and void sign(double y); have identical signatures - True
Constants must be declared public- False
A static variable is only visible **within** the class that defines it – False (it might be public)
(Note: this could have been misinterpreted from the original homework, True or False is OK in the answers. The original question should have been written as "within" instead of "with")
It is legal use the private access modifier to declare a temporary variable within the statement block that defines a method- False
If a java source file defines a public class called BigClass, the file must be called BigClass.java- True
It is legal to perform an assignment within a boolean expression- True
Variables of type int are automatically converted to doubles, when needed- True
Strings are a primitive type-False
A class can only define one constructor – False

7 why does the following moveTo method not do what is expected ?

```
Public void moveTo(double x,double y)
{
head.moveTo(x,y);
mouth.moveTo(x,y);
leftEye.moveTo(x,y);
rightEye.moveTo(x,y);
}
```

5 points for correct explanation of what code does

Above code does not do as expected because all the objects(head,mouth,leftEye and rightEye) move to specified location (x,y) with all their top left corners aligned(heads top left corner == mouth top left corner). We can modify the above code to function properly by adding appropriate offset to each part .For example leftEye.moveTo(x+15,y+15)

8 Assume gameOver is a boolean that is intialized to false. What is the difference between the ouput of the following two control structures ?

2.5 points for each correct answer
a
```
if(!gameOver)
    System.out.println("Roll again.");
}
```
This is an if statement ,so "Roll again" is printed once.

b
```
while(!gameOver){
    System.out.println("Roll again.");
 }
```
This is while loop and since the condition never becomes false, "Roll again" is printed infinite number of times.

10
Use DeMorgan's Laws to rewrite the following boolean expressions

a)(x == 25 || y < 45)
b)(delta < epsilon || i > 1000)

c)(charge < threshold && !hot)
d)(apple.color == Color.RED && leaf.color != Color.YELLOW)
e) (l.getEnd().getX() < 50 || time >= MAX_TIME)

1 point for each correct answer
a) !(x != 25 && y>=45)
b) !(delta >= epsilon && i<=1000)
c) !(charge > threshold || hot)
d) !(apple.color != Color.RED || leaf.color == Color.YELLOW)
e) !(l.getEn().getX() >= 50 && time < MAX_TIME)

11 Assume that x=6 , y =8 and z =-5. What are the values of x,y and z after the following code has been executed?
1 point for each correct answer to a), b) 3 points for correct braces in c)
a) if(x- (3+y)<=z-1)
x=y+2*z;
else if (x-2*y > 2*z)
y=z+y;
z=z+y;
x – (3+y) <= z-1
6 -(3+8) <= -5-1
5<=-6(FALSE)
so x=y+2*z is not executed

x-2*y> 2*z
6-2*8>2*-5
6-16>-10
-10>-10(FALSE)
so y=z+y is not executed

only z is update because of z=z+y;
-5+8=3

x=6 y=8 z=3

b) if(1-z >= 2*x – y)
y++;
x=x+y+z;

1-(-5)>=2*6-8
6>=4(TRUE)
y++ will update y
y=9
x=x+y+z will update x
x=6+9-5
x=10

x=10 y=9 z=-5

c)

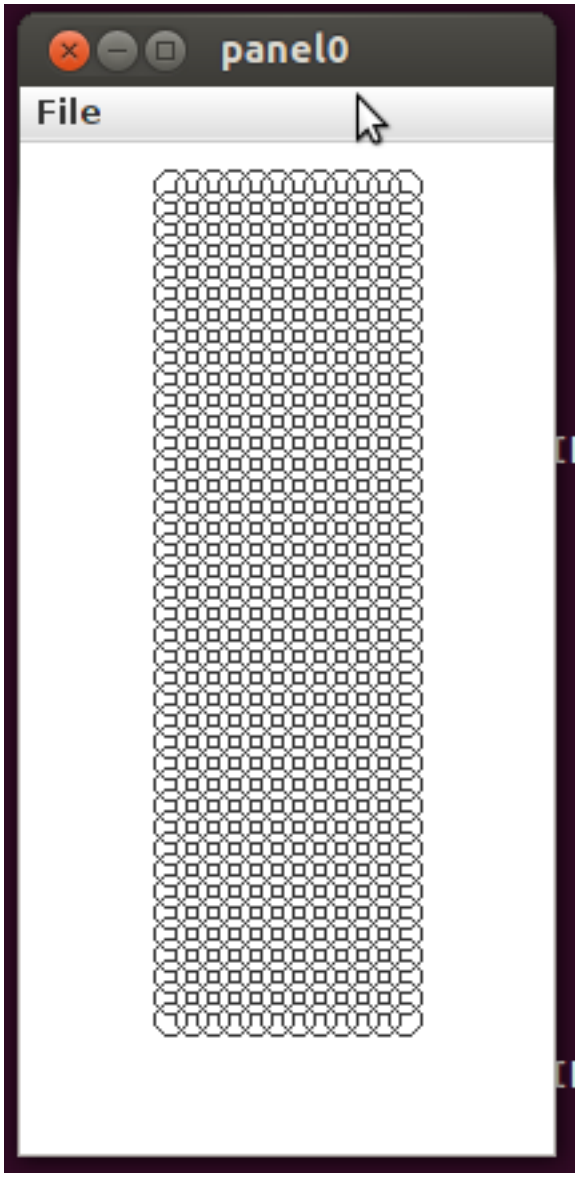
```
 if(x- (3+y)<=z-1)
{
      x=y+2*z;
}
else if (x-2*y > 2*z)
{
      y=z+y;
      z=z+y;
}

if(1-z >= 2*x – y)
{
      y++;
      x=x+y+z;
}
```

12 Using the NestedLoopGray.java program presented in lecture as a starting point, do JAEA 7.11.1. (This program is available from the class website)
Include code
Include output

5 points for correct (reasonable) code. 5 points for correct output

```
import objectdraw.*;
import java.awt.*;
public class KnitScarf extends WindowController

{
      private static final int WIN_SIZE = 400;
      private static final int DIAMETER = 12;
      private static final int rowNum=40;
      private static final int colNum=12;
      private static final int hue=50;
      private static final int upper_x=50;
      private static final int upper_y=10;
      private Text instructions;

      public void begin()
      {
            instructions = new Text("Click mouse to draw Circles", 50,WIN_SIZE-100, canvas);
      }
      public void onMouseClick(Location point)
      {
            instructions.hide();
            int row = upper_y;            // which row are we on
            while ( row < 10+(DIAMETER-4)*rowNum )
            {
                  int column = upper_x; // Start at the left
```

```
            while (column <50+ (DIAMETER-4)*colNum)
            {
                FramedOval circle;
                circle = new FramedOval(column, row,
                        DIAMETER,DIAMETER, canvas);
                circle.setColor(new Color(hue,hue,hue));
                column += (DIAMETER-4);

            }
            row += (DIAMETER-4); // Go to the next row
        }
    }


    public static void main(String[] args) {
        KnitScarf myWindow;
        myWindow = new KnitScarf();
        myWindow.startController(WIN_SIZE-200,WIN_SIZE);
    }
}
```



13. describe the following:

1. The difference between a public and private class variable

2. The difference between a private class variable and private static class variable
3.The difference between a private class variable and a local variable

1 point for correct answer to 1, 2 points each for correct answer to 2,3
1 Public class variable is accessible anywhere in the program

Private class variable is accessible only inside of the class in which it is declared.

2 private class variable is an instance variable and can be accessed only by a constructed instance of the class that defines it.

A private static class variable is a variable which can be accessed using <Classname>.<Variable> and the contents of this variable is shared among all instances of the class. (As an aside, these can be accessed by either constructed instances or from static methods of the class)

3 private class variables are visible inside all constructors and methods of the class in which they are declared.

Local Variables are visible only in the constructors or methods in which they are declared.

14. write a public method called sumSquares that returns an integer. SumSquares has two integer arguments called low and high. SumSquares computes the sum of the squares of the integers in the range [low,high]. Example: sumSquares (1,4) should return 1^2+ 2^2+ 3^2+4^2 = 30. You only need to write the method.

5 points for correct code

```
int sumSquares(int low,int high)
{
int result=0;
while(low<=high)
{
      result=result+(low*low);
      low++;
}
return result;
}
```

15 What does each of the java term mean ?

1 point for each correct definition

a. private-Private access modifier is the most restrictive access level.Methods, Variables and Constructors that are declared private can only be accessed within the declared class itself.

b. public Public access modifier has least restrictive access level.A class, method, constructor, interface etc declared public can be accessed from any other class.

c. local variable - A local variable in Java is a variable that's declared within the body of a method and can be used only within that method.

d. null- null is the reserved constant used in Java to represent a void reference i.e a pointer to nothing.

e. shadow– A variable is shadowed if there is another variable with the same name that is closer in scope. In other words, referring to the variable by name will use the one closest in scope , the one in the outer scope is shadowed.

16 A beginning programmer has written the program below without fully understanding declaration and scope. What problems does the following code have ?

```
Public class C{
    private int number ;

    public C(int aNumber){
        int myNumber=aNumber;
        int number =10;
}
...
private m()
{
...
if (myNumber > number){..}
}
...
}
```

2.5 points each for identifying an error in Constructor and method m. Each full explanation is not required for full credit, but the key issue (underlined) needs to be identified.

In the constructor C

int number is defined again in constructor C, so the 10 is stored in the variable number (local variable) which will be destroyed by end of constructor. (number is a shadow variable)

Value 10 should have been assigned to the instance variable number.

In method m()

myNumber is defined in constructor , so its scope is limited to constructor and destroyed at the end of constructor . So myNumber variable is not accessible in m( ) method. (saying the myNumber is undefined is an acceptable answer)

(number variable accessed in the m() method is the instance variable and not the temporary number variable used in the Constructor C.)