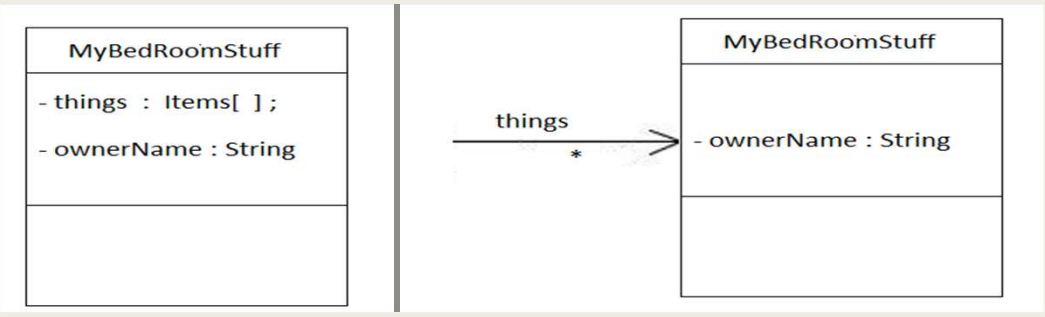
JAVA FINAL EXAM REVIEW SLIDES

v105 by Dave Slemon

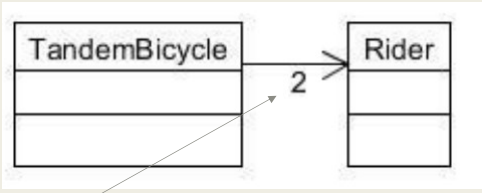
IMPORTANT UML Notation



These two UMLs are IDENTICAL.
- the one on the left displays the things array as an instance variable
- the one on the right displays the things array and it's unlimited size on top of the association arrow, BUT realize that the left diagram means the exact same as the right diagram.

IMPORTANT UML Notation

Read this as, "TandemBicycle has 2 Riders"




```
public class TandemBicycle
{
    Rider r1, r2;
}
```

```
public class Rider {

}
```

1. Static Class Variables

What is output to the terminal screen?

```
public class Locker
{
   private int lockerNumber;
   private static int nextId;

   public Locker()
   {
      if (nextId <= 0)
        nextId = 100;
      else
        nextId++;

      lockerNumber = nextId;
   }


   public String toString() {
      return String.format("%d",lockerNumber);
   }
}
```

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

      Locker a,b,c;

      a = new Locker();
      b = new Locker();
      c = new Locker();

      System.out.println(c);
   }
}
```

1. Static Class Variables

What is output to the terminal screen?

```
public class Locker
{
   private int lockerNumber;
   private static int nextId;

   public Locker()
   {
      if (nextId <= 0)
        nextId = 100;
      else
        nextId++;

      lockerNumber = nextId;
   }


   public String toString() {
      return String.format("%d",lockerNumber);
   }
}
```


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

      Locker a,b,c;

      a = new Locker();
      b = new Locker();
      c = new Locker();

      System.out.println(c);
   }
}
```

2. Switch Statement

Suppose: int day = 4;
Output the day number and the 3 letter day of the week using a switch statement, e.g. 0-Sun and 6-Sat.

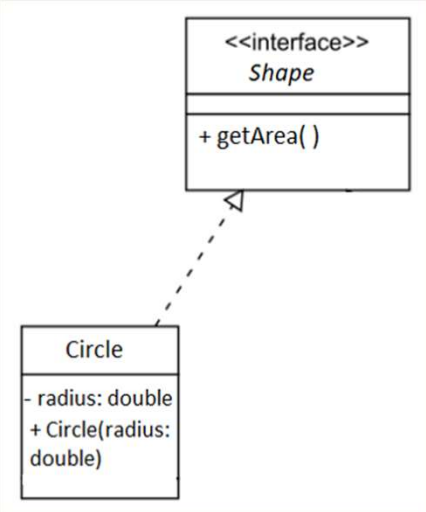
2. Switch Statement

Suppose: int day = 4;
Output the day number and the 3 letter day of the week using a switch statement, e.g. 0-Sun and 6-Sat.


```
int    day = 4;
String sDay = "";
switch (day) {
  case 0:
    sDay="Sun";
    break;
  case 1:
    sDay="Mon";
    break;
  case 2:
    sDay="Tue";
    break;
  case 3:
    sDay="Wed";
    break;
  case 4:
    sDay="Thu";
    break;
  case 5:
    sDay="Fri";
    break;
  case 6:
    sDay="Sat";
    break;
  default:
    sDay="Err";
}

System.out.println(day + "-" + sDay);
```

3. Interfaces

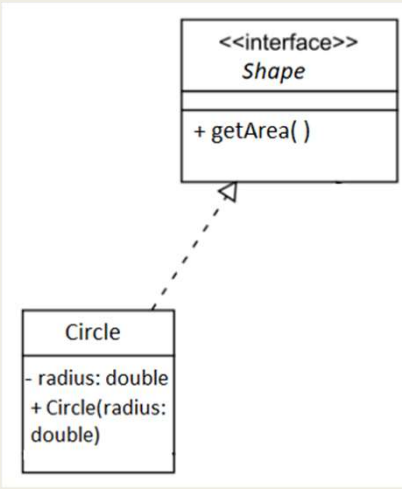
a) What's wrong with the UML diagram, what is the UML diagram missing?
a) Write the correct Java code for this diagram.



3. Interfaces

a) What's wrong with the UML?
b) Write the correct Java code for this diagram.

```
public interface Shape
{
    public double getArea( );
}
```



a) Compliance with the interface demands that the Circle class must have a public getArea( ) method

```
public class Circle implements Shape
{
    private double radius;

    public Circle(double radius) {
        this.radius = radius;
    }

    public double getArea( ) {
        return Math.PI * radius * radius;
    }
}
```

R4. ArrayList

Write the Java code which will declare and create an array called, numbers which holds the numbers: 4.5, 5.5 and 6.5. Output the array as shown on the terminal window.

4. ArrayList

Write the Java code which will declare and create an arraylist called, numbers which holds the numbers: 4.5, 5.5 and 6.5. Output the array as shown on the terminal window.


```
import java.util.ArrayList;

public class TestProgram
{
   public static void main(String[] args) {
      ArrayList<double> numbers = new ArrayList<>();

      numbers.add(4.5);
      numbers.add(5.5);
      numbers.add(6.5);

      System.out.println(numbers);
   }
}
```

Oops!! ...there's something wrong with this answer.
CAN YOU FIND THE ERROR?

4. ArrayList

Write the Java code which will declare and create an arraylist called, numbers which holds the numbers: 4.5, 5.5 and 6.5. When the program is run the output is shown on the terminal window.


```
import java.util.ArrayList;

public class TestProgram
{
   public static void main(String[] args) {
      ArrayList<Double> numbers = new ArrayList<>();

      numbers.add(4.5);
      numbers.add(5.5);
      numbers.add(6.5);

      System.out.println(numbers);
   }
}
```

MAJOR ISSUE:
In Java, ArrayList stores object references rather than the actual objects themselves. This applies to all Java collections.

When you add an element to an ArrayList, it stores a reference to the object you added, not a copy of the object's data. YOU CAN NOT STORE primitive ty like, *int, double, boolean* or *char* in an ArrayList. Use their corresponding primitive wrapper instead! (i.e. Integer, Double, Boolean or Char)

5. Catching and Throwing Unchecked Exceptions

Study the program below, it crashes with an ArrayIndexOutOfBoundsException. It does however display "Error: index out of bounds" to the terminal window before it crashes. CAN YOU FIX THE PROGRAM SO IT DOESN'T CRASH, and STILL DISPLAYS THE SAME ERROR TO THE TERMINAL WINDOW?

```
public class ExceptionExample {
    public static void main(String[] args) {
        int[] numbers = { 1, 2, 3, 4, 5 };
        int index = 10;
        int result = getElementAt(numbers, index);
        System.out.println("Element at index " + index + ": " + result);
    }

    public static int getElementAt(int[] arr, int index) {
        if (index < 0 || index >= arr.length) {
            System.out.println("Error: index out of bounds");
        }
        return arr[index];
    }
}
```

you will need to catch the exception

you're going to have to remove This line and replace it with something else

5. Catching and Throwing Unchecked Exceptions

Study the program below, it crashes with an ArrayIndexOutOfBoundsException. It does however display "Error: index out of bounds" to the terminal window before it crashes. CAN YOU FIX THE PROGRAM SO IT DOESN'T CRASH, and STILL DISPLAYS THE SAME ERROR TO THE TERMINAL WINDOW?

```
public class ExceptionExample {
    public static void main(String[] args) {
        try {
            int[] numbers = { 1, 2, 3, 4, 5 };
            int index = 10;
            int result = getElementAt(numbers, index);
            System.out.println("Element at index " + index + ": " + result);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Caught an exception: " + e.getMessage());
        }
    }

    public static int getElementAt(int[] arr, int index) {
        if (index < 0 || index >= arr.length) {
            throw new ArrayIndexOutOfBoundsException(" Error: index out of bounds");
        }
        return arr[index];
    }
}
```

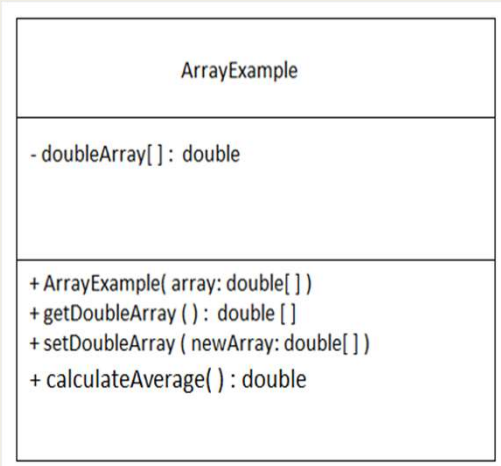
CATCH EXCEPTION

THROW EXCEPTION

6. Built-In Arrays & Going from UML->Code

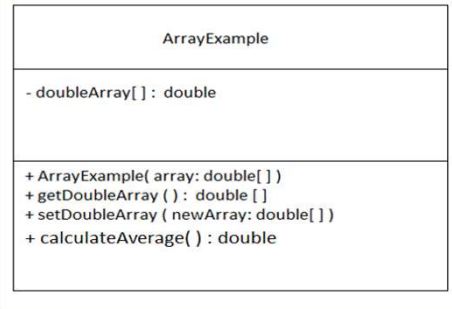
Code the following class into Java so that the main pgm below works.

(no need for exceptions)




```
public static void main(String[] args) {
    double[] exampleArray = { 3.5, 2.0, 9.8, 1.2, 6.4 };

    ArrayExample example = new ArrayExample(exampleArray);

    System.out.println("Original Array:");
    for (double num : example.getDoubleArray()) {
        System.out.print(num + " ");
    }
    System.out.println("\nAverage: " + example.calculateAverage(),,
}
```

6. Built-In Arrays & Going from UML->Code




```
public class ArrayExample {
    private double[ ] doubleArray;

    public ArrayExample(double[ ] array) {
        // Constructor to initialize the instance variable
        doubleArray = array;
    }

    public double[ ] getDoubleArray() {
        return doubleArray;
    }

    public void setDoubleArray(double[ ] newArray) {
        doubleArray = newArray;
    }
```

```
    public double calculateAverage() {
        // Calculate and return the average
        if (doubleArray.length == 0) {
            return 0.0; // Handle the case when the array is empty
        }

        double sum = 0.0;
        for (double num : doubleArray) {
            sum += num;
        }
        return sum / doubleArray.length;
    }
```

7. One-liners

a. What's the line that a button, called GoButton uses to trigger a handler routine called, goButtonTask

```
GoButton.setOnAction( this:: goButtonTask );
```

b. What's the line that declares and populates a 1-d built-in integer array with values, 3, 9 & 27

```
int[ ] numbers = { 3, 9, 27 } ;
```

c. What's the line that triggers a routine called, pressHander to run in a canvas called, "c", after the primary mouse button is pressed?

```
c.addEventHandler(MouseEvent.MOUSE_PRESSED, this::pressHandler);
```

d. What's the line which will declare an String arraylist called, A.

```
ArrayList<String> A = new ArrayList<String>( );
                or
ArrayList<String> A = new ArrayList<>( );
```

7. One-liners

e. What's the first line of a method handler, called goButtonTask, which will run if triggered by the line
GoButton.setOnAction(this:: goButtonTask);

```
public void goButtonTask ( ActionEvent e )  {

}
```

f. What's the first line of the method hander, called pressHander, which will run if triggered by
canvas.addEventHandler(MouseEvent.MOUSE_PRESSED, this::pressHandler);

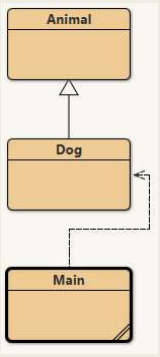
```
public void pressHandler( MouseEvent me ) {

}
```

g. What's the if statement condition, when you compare two strings, s1 and s2 for equality. Why can't you use == sign?

```
if  ( s1.equals(s2) ) {
   //you can't use the == because  s1 and s2 are reference addresses that point to the
   // string values,  Always use a method when comparing two objects.
}
```

8. super( ) and this ( ) - Inheritance

Study the code below, and write the **Dog class**, so that the product shot is realized exactly when the main method below is run.



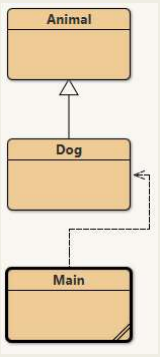

```java
class Animal {
    private String name;

    // Constructor of the superclass
    Animal(String name) {
        this.name = name;
        System.out.println("Animal constructor called");
    }

    void makeSound() {
        System.out.println("Animal makes a sound");
    }

    public String getName() {
        return name;
    }
}
```

```java
public class Main {
    public static void main(String[] args) {
        Dog myDog = new Dog("Kato", "German Shepherd");
        Dog friendsDog = new Dog("Rocky");
        System.out.println("My Dog: " + myDog);
        System.out.println("My Friend's Dog: " + friendsDog);
        myDog.makeSound();
    }
}
```

```
Animal constructor called
Dog constructor called
Animal constructor called
Dog constructor called
My Dog: Kato is a German Shepherd
My Friend's Dog: Rocky is a Mutt
Kato barks
```

8. super( ) and this ( ) - Inheritance

Study the code below, and write the **Dog class**, so that the product shot is realized exactly when the main method is run.



Code the most complicated constructor first, i.e. the one with the most parameters

...then just call the most complicated Constructor; this will avoid duplicate code

```java
class Dog extends Animal {
    private String breed;


    //default constructor
    Dog () {
        this("Doggy","Mutt");
    }

    //avoids repeating
    Dog(String name) {
        this(name,"Mutt");
    }


    // Constructor of the subclass, calling the superclass constructor using super()
    Dog(String name, String breed) {
        super(name); // Calling the constructor of the superclass
        this.breed = breed;
        System.out.println("Dog constructor called");
    }


    public void makeSound() {
        System.out.println(getName() + " barks");
    }


    public String toString() {
        return getName() + " is a " + breed;
    }
}
```

```
Animal constructor called
Dog constructor called
Animal constructor called
Dog constructor called
My Dog: Kato is a German Shepherd
My Friend's Dog: Rocky is a Mutt
Kato barks
```

9a. For Loops

Study the main program below and write from scratch the Employee class and then display to the terminal windows the list of employees and their current hourly wage in the EXACT format below..

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

  //Creating a list of employees
  ArrayList<Employee> myPeople =new ArrayList<>();
  myPeople.add(new Employee("Jin",15.00));
  myPeople.add(new Employee("Bashshar",17.00));
  myPeople.add(new Employee("Chris",19.00));


  //traversing the employees using an enhanced for loop
  for(Employee e : myPeople){
    System.out.println(e);
  }

 }
}
```

9a. For Loops

Study the main program below and write from scratch the Employee class and then display to the terminal windows the list of employees and their current hourly wage in the EXACT format below..

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

  //Creating a list of employees
  ArrayList<Employee> myPeople =new ArrayList<>();
  myPeople.add(new Employee("Jin",15.00));
  myPeople.add(new Employee("Bashshar",17.00));
  myPeople.add(new Employee("Chris",19.00));


  //traversing the employees using an enhanced for loop
  for(Employee e : myPeople){
    System.out.println(e);
  }

 }
}
```

```
public class Employee {
   private String name;
   private double wage;

   public Employee(String name, double startingWage) {
      this.name = name;
      this.wage = startingWage;
   }


   public String toString() {
      return String.format("%-10s $%.2f / hr",name,wage);
   }
}
```

9b. For Loops (using only ENHANCED for loops – answering this question.)

i) Add a method to Employee to allow for wage increases, i.e. payRaise(double amount) .
ii) In the main program, give everyone a 50 cent raise.

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

  //Creating a list of employees
  ArrayList<Employee> myPeople =new ArrayList<>();
  myPeople.add(new Employee("Jin",15.00));
  myPeople.add(new Employee("Bashshar",17.00));
  myPeople.add(new Employee("Chris",19.00));


  //traversing the employees using an enhanced for loop
  for(Employee e : myPeople){
    System.out.println(e);
  }

 }
}
```

```
public class Employee {
   private String name;
   private double wage;

   public Employee(String name, double startingWage) {
      this.name = name;
      this.wage = startingWage;
   }


   public String toString() {
      return String.format("%-10s $%.2f / hr",name,wage);
   }
}
```

9b. For Loops (using only ENHANCED for loops – answering this question.)

i) Add a method to Employee to allow for wage increases, i.e. payRaise(double amount) .
ii) In the main program, give everyone a 50 cent raise.

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

  //Creating a list of employees
  ArrayList<Employee> myPeople =new ArrayList<>();
  myPeople.add(new Employee("Jin",15.00));
  myPeople.add(new Employee("Bashshar",17.00));
  myPeople.add(new Employee("Chris",19.00));

  for(Employee e : myPeople){
    e.payRaise(0.50);
  }

  //traversing the employees using an enhanced for loop
  for(Employee e : myPeople){
    System.out.println(e);
  }

 }
```

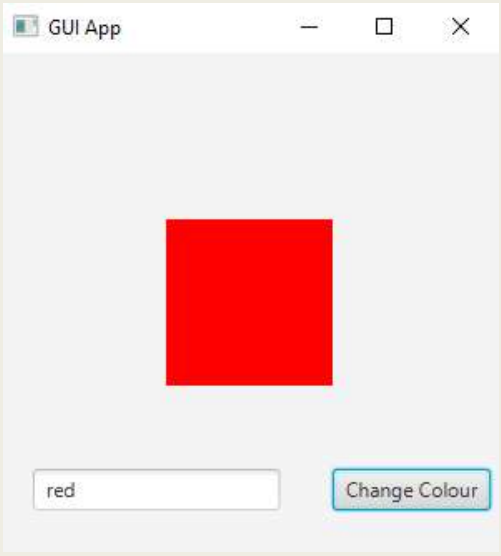
```
public class Employee {
   private String name;
   private double wage;

   public Employee(String name, double startingWage) {
      this.name = name;
      this.wage = startingWage;
   }

  public void payRaise(double amount) {
      wage = wage + amount;
   }

   public String toString() {
      return String.format("%-10s $%.2f / hr",name,wage);
   }
}
```

10. GUI programming

Code into Java this GUI App.

Use the standard Java class Rectangle to handle the rectangle.

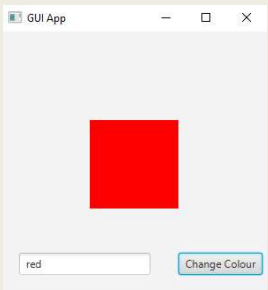
```
import javafx.scene.shape.Rectangle;
```




```
public void start(Stage stage) throws Exception {
      Pane root = new Pane();
      Scene scene = new Scene(root, 300, 300);
      stage.setTitle("GUI App");
      stage.setScene(scene);

      // Create the model
     // Specify x, y, width, and height
      rectangle = new Rectangle(100, 100, 100, 100);
```

10. GUI programming

```
public class GUIApp extends Application {

    private Rectangle rectangle;
    private GraphicsContext gc;
    private TextField inputBox;
    private Button button;

    public void colorHandler( ActionEvent e) {
       String str = inputBox.getText();

       rectangle.setFill(  Color.valueOf(str) );

    }
```




```
public void start(Stage stage) throws Exception {
      Pane root = new Pane();
      Scene scene = new Scene(root, 300, 300);
      stage.setTitle("GUI App");
      stage.setScene(scene);

      // Create the model
      rectangle = new Rectangle(100, 100, 100, 100);

      // Create the GUI components
      Canvas c = new Canvas(300, 300);
      gc = c.getGraphicsContext2D();
      button = new Button("Change Colour");
      inputBox = new TextField("red");
       // Add components to the root
      root.getChildren().addAll(c, rectangle,button,inputBox);

      // Configure the components (colors, fonts, size, location)
      rectangle.relocate(100, 100);
      rectangle.setFill(Color.BLUE); // Set fill color
      inputBox.relocate(20,250);
      button.relocate(200,250);

      // Add Event Handlers and do final setup
      button.setOnAction( this::colorHandler);
      // Show the stage
      stage.show();
}
```

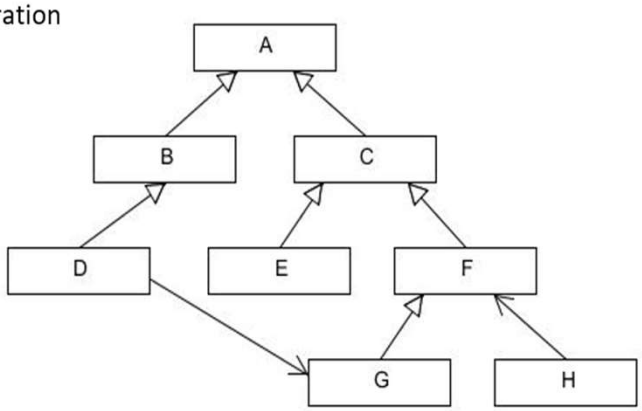
Be sure you can write the code shown in green.

11 - Inheritance

Given the UML diagram and the declaration for the variable c below, put a check mark beside all the assignments that are legal and an X beside those that are illegal. [3]

```
C c;
```
← This is the variable declaration

```
c = new A();
c = new B();
c = new C();
c = new D();
c = new E();
c = new F();
c = new G();
c = new H();
```

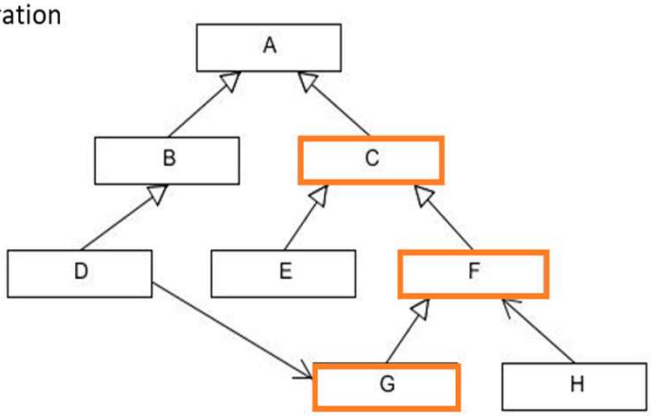


11 - Inheritance

Given the UML diagram and the declaration for the variable c below, put a check mark beside all the assignments that are legal and an X beside those that are illegal. [3]

```
C c;
```
← This is the variable declaration

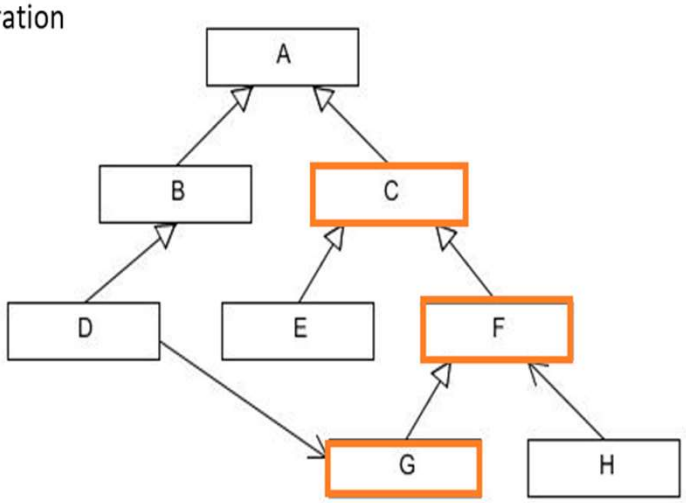
```
c = new A(); X
c = new B();
c = new C();
c = new D();
c = new E();
c = new F();
c = new G();
c = new H();
```



11 - Inheritance

C c; ← This is the variable declaration

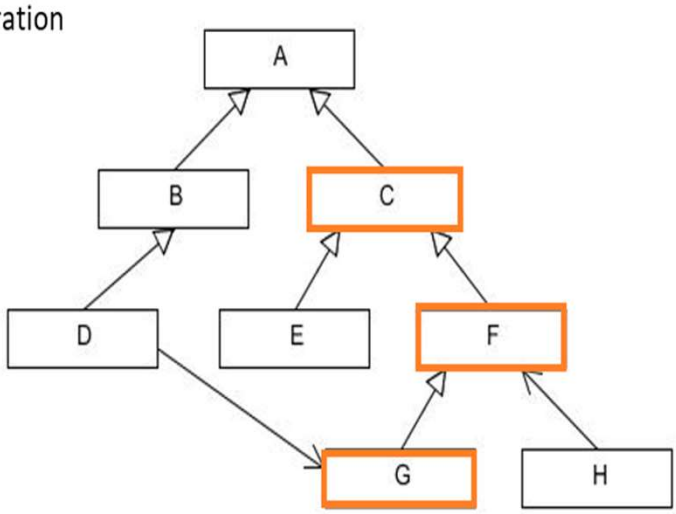
```
c = new A(); X
c = new B(); X
c = new C(); ✓
c = new D(); X
c = new E();
c = new F();
c = new G();
c = new H();
```



11 - Inheritance

C c; ← This is the variable declaration

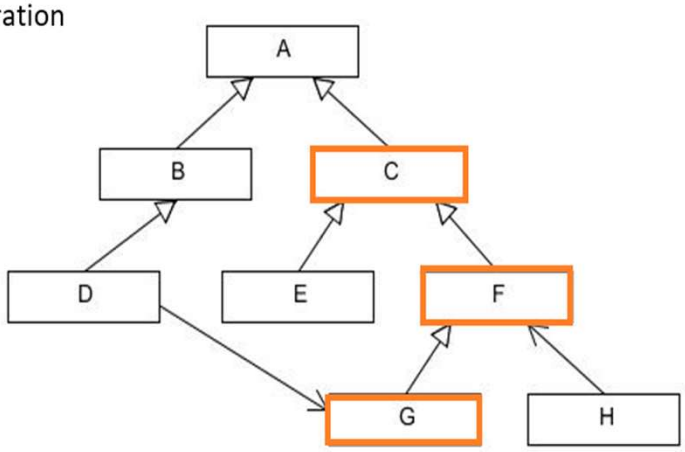
```
c = new A(); X
c = new B(); X
c = new C(); ✓
c = new D(); X
c = new E(); ✓
c = new F(); ✓
c = new G(); ✓
c = new H();
```



11 - Inheritance

C c; ← This is the variable declaration

```
c = new A(); X
c = new B(); X
c = new C(); ✓
c = new D(); X
c = new E(); ✓
c = new F(); ✓
c = new G(); ✓
c = new H(); X
```



12. Write the java code for a *setRadius(int radius)* method Which accepts only a POSITIVE integer value greater than ZERO for the radius. If the radius supplied is less than or equal to 0, throw an BadRadiusException with a message, "Radius must be >= 0"

12. Write the java code for a *setRadius(int radius)* method Which accepts only a POSITIVE integer value greater than ZERO for the radius. If the radius supplied is less than or equal to 0, throw an BadRadiusException with a message, "Radius must be >= 0"

```
public void setRadius(int radius)
{
    if (radius <= 0) {
        throw new BadRadiusException("Radius must be >= 0")
        this.radius = radius;
    }
}
```

13. Write a main program which uses the Scanner to ask the user for his/her age. Store the age received in a integer variable called, age. If the user enters a string instead for the age, assign the age to 0 and output an appropriate error message to the terminal window. . (use InputMismatchException)

13. Write a main program which uses the Scanner to ask the user for his/her age. Store the age received in a integer variable called, age. If the user enters a string instead for the age, assign the age to 0 and output an appropriate error message to the terminal window. (use InputMismatchException)

```
public static void main(String [ ] args) {
    int age = 0;
    Scanner keybd = new Scanner(System.in);
    try
    {
      age = keybd.nextInt( )
      keybd.nextLine(); //clear the line
    }
    catch ( InputMisMatchException e )
    {
      System.out.println("Error: age must be >= 0");
      age = 0;
    }

}
```

14. Write code that will create an array of 1000 Counter objects called, myCounters. Next, increment each counter by a random integer between 0 and 100 inclusive. Then, display the current count of each counter using an enhanced for loop. IMPORTANT: you may **NOT** add or change the Counter class in anyway!

```
/**
  Provides a random int between minimum and maximum,not including maximum
  @return minimum <= num < maximum
*/
  public static int getRandomInteger( int minimum, int maximum)
  {
    return ((int) (Math.random() * (maximum - minimum))) + minimum;
  }
```

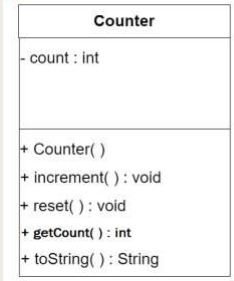
Counter class shown below. **Note: When a new Counter is constructed its count is set to zero**

Counter
- count : int
+ Counter( ) + increment( ) : void + reset( ) : void + getCount( ) : int + toString( ) : String

14. Write code that will create an array of 1000 Counter objects called, myCounters. Next, increment each counter by a random integer between 0 and 100 inclusive. Then, display the current count of each counter using an enhanced for loop. IMPORTANT: you may **NOT** add or change the Counter class in anyway!

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

    Counter[ ] myCounters = new Counter[1000];

    for(int c=0; c< myCounters.length; c++)
       myCounters[c] = new Counter();

    int randomNumber;

    for(int c=0; c< myCounters.length; c++) {

        randomNumber = getRandomInteger(0,101);

        for(int x=0; x < randomNumber; x++)
            myCounters[c].increment();
    }


    for ( Counter count : myCounters)
       System.out.println ( count.getCount() + " ");


}
```

Counter class shown below. **Note: When a new Counter is constructed its count is set to zero**



15. Write a INTERFACE class which demands that all classes compliant have a

getArea() method.

15.Write a INTERFACE class which demands that all classes compliant have a
getArea() method.

```
public interface measureable (  ) {

    public double getArea();

}
```

16.(following on from question 15), give an example of the header line belonging to a class that obeys, measurable

```
public interface measureable (  ) {

    public double getArea();

}
```

16.(following on from question 15), give an example of the header line belonging to a Circle class that obeys, measurable

```
public class Circle implements measurable {

   //etc

}
```