CS100J 03 Dec 2003 Sample questions (with answers at the end)

Questions on OO

OO1. Consider the classes below and write code to complete the class SportsCar

```
public class Car{
    private double milesPerGallon;

    // = the miles per gallon for this car
    public Car(double mpg){
        milesPerGallon = mpg;
    }

    // = a string representation of this car
    public String toString(){
        return "This car gets " + milesPerGallon + " miles per gallon.";
    }
}

public class SportsCar extends Car{
    private int topSpeed;

    // Constructor: an instance with miles per gallon mpg and top speed topSpeed
    public SportsCar(double mpg, int topSpeed){
        //Write code to complete the constructor here
    }

    /* = A string that gives the fuel economy and top speed of
    this sports car.  Hint:  This string will be more than one sentence.  */
    public String toString(){
        //Write your code here
    }

    // = the top speed of this car
    public int getTopSpeed(){
        return topSpeed;
    }
}
```

OO2. (a) With reference to the class Car in OO1, draw the folder that results from the call

```
    Car myCar= new Car(25.5);
```

(b) With reference to the picture you just drew, during execution of the following statement, what would be the value of the variable **'this'** if it occurred in method toString?

```
    myCar.toString();
```

(c) True or False: Variable topSpeed can be referenced in class SportsCar.

OO3. With reference to the Car and SportsCar classes of OO1, consider the following statements. Next to each statement write whether it is a legal statement or not.

```
    SportsCar sc= new SportsCar(15.0, 170);
  Car mySportsCar= sc;
  int ts= mySportsCar.getTopSpeed();
  double ts2= sc.getTopSpeed();
  Car c= new Car(true, "Red");
  Car myOtherCar= new Car(45.2);
  SportsCar sc2= myOtherCar;
```

OO4. Consider class PrintedMaterial:

```
public abstract class PrintedMaterial{
    public int numPages;
    public String printedLanguage;
    public abstract String getPrintedLanguage();
}
```

(a) If you write a class Book to extend PrintedMaterial:

```
        public class Book extends PrintedMaterial{
            private String title;
            private String author;

            //there are no other fields, but the rest of the details are omitted...
}
```

can you say anything about the methods that you must include?

(b) Write a constructor for class Book whose arguments are the number of pages, the printed language, the title, and the author.

(c) Next to the following statements, write whether they are legal or not:

```
Book b= new Book(301, "English", "The Brothers Karamazov", "Dostoyevsky");
PrintedMaterial pm= b;
PrintedMaterial pm2= new PrintedMaterial();
PrintedMaterial[] pmArray= new PrintedMaterial[9];
```

OO5. An online retail store sells shirts, pants, and jackets. Below are two classes representing RetailItems and Shirts.

(a) Fill in the code for method putOnSale().

(b) (b) fill in the code of class Shirt:

c) For each of the following statements, answer the questions that appear indented under the statements. Assume that the statement(s) in each part is (are) written in a main method in a class called Test and are executed independently of other parts of the question.

Shirt s = **new** Shirt(45.50, “Land’s End”, “L”);
What is/are the apparent type of r? ________________________
What is/are the real type of r? ________________________

RetailItem t = **new** Shirt(80.50, “Ralph Lauren”, “M”);
What is/are the apparent type of t? ______________________
What is/are the real type of t? ________________________

RetailItem t = **new** Shirt(80.50, “Ralph Lauren”, “M”);
t.getSize();
Will the second statement produce a compiler error (Yes or No)? _____________
Why or why not? ____

RetailItem t = **new** Shirt(80.50, “Ralph Lauren”, “M”);
t.toString();
Will the second statement produce a compiler error (Yes or No)? ______________
If yes, what is the error? If no, what String will be returned?

```
/** An instance represents a retail item */
public class RetailItem {
        private double price; // price of item
        private String manufacturer; // manufacturer of item

        /** Constructor: a RetailItem with given price and manufacturer */
        public RetailItem(double price, String manufacturer) {
                this.price = price;
                this.manufacturer = manufacturer;
```

```
    }

    /** = the price of this item */
    public double getPrice() { return price; }

    /** = the manufacturer of this item */
    public String getManufacturer() { return manufacturer; }

    /** put item on sale by reducing cost by percentage given;
       return true if successful and false otherwise */
    public boolean putOnSale(double percentage) {
        if (0.0 < percentage && percentage < 1.0) {
            price = price * (1.0 – percentage);
            return true;
        }
        return false;
    }

    /** put item on sale by reducing cost by 50 percent */
    public boolean putOnSale() {
      // put code here
    }

    /** = a String representation of this item */
    public String toString() {
        return “RetailItem[price = “ + price + “, manufacturer = “ + manufacturer + “]”;
    }
}

/** An instance represents a Shirt */
public class Shirt extends RetailItem {
    private String size; // size of shirt – S, M, L, XL

    /** Constructor: a Shirt item with given price, manufacturer, and size*/
    public Shirt(double price, String manufacturer, String size) {
        // fill in code
    }

    /** = the size of this shirt*/
    public String getSize() {
      // fill in code
    }

    /** = a representation of the shirt */
    public String toString() {
        // fill in code
    }
}
```

OO6. You are working for an online retail store that sells clothing. Your manager has asked you to modify class Customer so that it assigns a new unique customer ID number when a customer object is created. The current implementation constructs a new Customer object using the ID number supplied as a parameter.

Your manager no longer trusts client code to assign unique ID numbers when creating Customer objects. **Show the changes** that you need to make to the existing class Customer below so that each new Customer object has a unique ID number. The ID number of the first newly created customer object should be 1, the next customer object created should have ID number 2, and so on.

```
/** An instance represents a Customer */
public class Customer {
   private int idNumber; // customer ID number
   private String name; // customer name
```

```
    private ShoppingCart items; // customer's shopping cart
    private Address mailingAddress; // customer's mailing address

    /** Constructor: a new customer object with idNumber, name, and
       mailing address */
    public Customer(int idNumber, String name, Addres mailingAddress) {
          this.idNumber= idNumber;
          this.name=name;
          this.mailingAddress= mailingAddress;
          this.items= new ShoppingCart();
    }
}
```

OO7. Consider class Movie and its two subclasses Comedy and Drama, given below. Consider the following piece of code, with lines numbered, which contains some illegal statements.
(a) Which statements would cause a compile-time error (write down the line numbers).
(b) Pretend we compile and run the remaining code. Mark which statements, if any, result in a runtime exception.
(c). Pretend we ignore all the code that causes either a compile-time error or a runtime exception. If we compile and run the program, what gets printed in the Java console?

```
(1)  Comedy rushHour= null, officeSpace= null, theSantaClause= null;
(2)  Movie shawshank= null;
(3)  Drama braveheart= null;

(4)  rushHour= new Movie("PG-13");
(5)  officeSpace= new Comedy("R");
(6)  theSantaClause= new Drama("PG");
(7)  shawshank= new Drama("R");
(8)  braveheart= new Drama("R");

(9)  System.out.println(rushHour.getRating());
(10) System.out.println(officeSpace instanceof Movie);
(11) System.out.println(theSantaClause == officeSpace);
(12) System.out.println(shawshank.getResponse());
(13) System.out.println(shawshank.getResponse("Calvin"));
(14) System.out.println(braveheart.getResponse("Hobbes"));
(15) System.out.println(braveheart instanceof Object);
(16) System.out.println(braveheart.getRating());
```

```
public class Movie {
   private String rating;

   public Movie(String rating) {
      this.rating= rating;
   }

   public String getRating() {
      return rating;
   }

   /**  = the emotional response to this Movie */
   public String getResponse() { return "no response"; }
}

public class Comedy extends Movie {
   public Comedy(String rating) { super(rating); }

   public String getResponse() { return "laughter";}
}

public class Drama extends Movie {
   public Drama(String rating) { super("NC-17"); }

   /** = the emotional response of person to this Drama */
   public String getResponse(String person) {
      if (person.equals("Calvin")) return "boredom";
      else return "tears";
   }
}
```

OO8. Consider classes TVChannel and FOX, as given below.
(a) Which fields declared in TVChannel are accessible in FOX?
(b) Which methods declared in TVChannel are accessible in FOX?
(c) Name two methods NOT declared in TVChannel that are accessible in FOX.

```
public class TVChannel {
      private int channelNumber;
      private String network;
      private double frequency;

      public TVChannel(int num, String net) {
            channelNumber= num;
            network= net;
            frequency= 1.0 / channelNumber;
      }

      public double getFrequency() { return frequency; }

      private StringBuffer toStringBuffer() { return new StringBuffer(network); }

      public int toInt() {  return channelNumber; }
}

public class FOX extends TVChannel {
      public FOX(int channelNumber, String network) { super(channelNumber, "FOX"); }
}
```

OO9. 3. Consider classesVehicle and Car, shown below. Vehicle contains 4 errors and Car contains 3, where an error is defined as either a syntactic error or a statement that doesn't do what it’s supposed to do. Write down what these errors are.

```
abstract class Vehicle {
      private int numWheels;  //number of wheels in the vehicle
      String model;                   //the model

      // Constructor: an instance with wheels wheels and is model mod
      public Vehicle(int wheels, String mod) {
            numWheels= wheels;
            model== mod;
      }

      public int getNumWheels() { int result= numWheels; }

      public String getModel() { return model }

      public static void main(String[] args) Vehicle v= new Vehicle(4, "GS300"); }
}

// An instance is a car
public class Car extends Vehicle {
      private int numDoors; // number of doors
```

```
   // Constructor: car with wheels wheels, model mod, and number of doors numDoors
    public Car(int wheels, String model, int numDoors)
         { numDoors= numDoors; super(wheels, model); }

    public static void main(String[] args) { Car c1= new Car(4, "Maxima SE", 4);   Car c2= new Car(); }
}
```

OO10. Consider classes Country and USA, shown below.
(a) Write the constructor for class Country.
(b) Write the constructor for class USA.
(c) Write method getPopulation() in class Country.
(d) Write method changePopulation(long) in class Country, which increases the population by some amount. Without adding any new methods or fields, how could you use method changePopulation to decrease the population?
(e) Write method printInfo() in class USA.
(f) Draw a folder for class USA.

```
public class Country {
    private long population; // the population in this country

    /** Constructor: An instance of country that has the specified population.       */
    public Country(int population) {                                              }

    /** = the population of this country */
    public long getPopulation() {                                                 }

    /** Add increase to the current population */
    public void changePopulation(long increase) {                                                  }
}

public class USA extends Country {

    /** Constructor: A country with initial population 222,000,000   */
    public USA() {                                                                  }

    /** Print to the console a String of the format  "USA's population: <population>" */
    public void printInfo() {                                                        }
}
```

OO11. Below there are two classes defined, Boy and GoodBoy, and a piece of code using them. Something is wrong with these definitions and/or the code referring to them, however. Point to three major mistakes made by the programmer and suggest how one could fix them, if possible, without significantly changing functionality and while preserving the overall structure of the code.

```
// An instance is a boy with a name
public abstract class Boy {
    private String myname;  // name of boy

    // Constructor: an instance with name myname
    public Boy(String myname { this.myname = myname; }

    // Print a message
    public abstract void sayHello() {
        System.out.println("Hello, I'm " + myname + ", a boy.");
    }
}

// An instance is a boy with a name
public class GoodBoy extends Boy {

    // Constructor: an instance with name myname
    public GoodBoy(String name) { super(name); }

    // Print a message
    public void sayHello() {
        System.out.println("Hello, I'm " + myname + ", a good boy.");
    }
}

// This is typed in DrJava's interactions pane
Boy b= new Boy("John");
GoodBoy g= new GoodBoy("Adam");
b= g;
```

OO12. Define an abstract class ChristmasGift, subclasses of which will represent various kinds of gifts, all of which can be compared to each other. As in real life, one can compare not only gifts of the same type but also gifts of different kinds. Make sure that ChristmasGift is defined appropriately so that its subclasses can be compared to each other. Avoid anything unnecessary beyond what is essential for the subclasses to be able to be compared. You also need to write two subclasses NewPairOfSocks and GiftCertificate, the first of which does not have any attributes and the second of which has only a single **int** field representing its value in dollars. Add appropriate methods so as to ensure that, when compared, gift certificates of higher value report that they are better than gift certificates of lower value, and a gift certificate is better than socks as long as its value exceeds 20 dollars. Also, both gift certificates and
socks are better than any other kind of Christmas gift.

QUESTIONS ON THE EXECUTION MODEL

EX1. Use method isPalindrome below to answer the following question.

```
    public class Word {
        /** = "word[start..end] is a palindrome" –i.e. it is its own reverse.
                Precondition: start <= end, start >= 0, end < word.length
        public static boolean isPalindrome(char[] word, int start, int end) {
            if (start >= end)  return true;
            if (word[start] != word[end]) return false;
            return isPalindrome(word, start+1, end–1);
        }
    }
```

Assume that character array racecar has been initialized to the following in the Dr. Java interactions window:

```
        char[] racecar = {'r', 'a', 'c', e', c', 'a', r'};
```

Draw the model of execution showing what happens during execution of the following method call in the Dr. Java interactions window:

```
        boolean racecarPalindrome = Word.isPalindrome(racecar, 0, racecar.length-1);
```

EX2. Consider class Blah, shown below. Write down what appears on the screen after method main() is completed. You are not required to draw folders and all, but if you don't, you are quite likely to get it all wrong.

```
import javax.swing.*;
public class Blah {
    private JFrame frame;

    public void main() {
        JFrame x= new JFrame("Jesse");
        JFrame y= new JFrame("Tulip");
        JFrame z= new JFrame("Cass");
        frame= y;

        foo(x);
        bar(y);
        blah(z);

        frame= new JFrame(y.getTitle());
        y= x;
        JFrame w= new JFrame(y.getTitle());

        System.out.println(x.getTitle());
        System.out.println(y.getTitle());
        System.out.println(z.getTitle());
        System.out.println(w.getTitle());
        System.out.println(frame.getTitle());
        System.out.println(w == y);
    }

    public void foo(JFrame frame) { frame= new JFrame("Foo"); }

    public void bar(JFrame frame) { this.frame= frame; frame.setTitle("Bar"); }

    public void blah(JFrame frame) { this.frame= frame; this.frame.setTitle("Blah"); }
        }
```

EX3. Consider class Elbor below. Execute it and write down what appears on the screen after method main() is completed. You are not required to draw the folders and frames, but you are quite likely to get the answer from if you don't.

```
public class Elbor {
    int i;

    public Elbor(int i) { System.out.println("constructor"); this.i= i; }

    public String toString() { return String.valueOf(i); }

    public static void main() { System.out.println(new Elbor(getSqrt(1) + getCube(1))); }

    public static int getSqrt(int num) { System.out.println("sqrt"); return getInt(Math.sqrt(num)); }

    public static int getInt(double d) { System.out.println("int"); return (int) d; }

    public static int getCube(int num) {
        int cube= num*num*num;
        System.out.println("cube");
        return cube;
    }
}
```

EX4. Consider the following class:

```
    public class A {
        public int x;

        public A(int z) {  x= z; }
}
```

What is printed by execution of the following code?

```
    A z= new A(4);   A y= new A(9);
    z.x= 10;    y= z;
    y.x= 11;   z.x= z.x * 10;
    System.out.println(y.x);
```

EX5. Consider the class of EX4 and the following class:

```
    public class B extends A {
        public int y;

        public B(int w1, int w2)
            { super(w1 * 10); y = w2; }
}
```

What is printed by execution of the following code?

```
    A a1= new B(4,5);  B a2= a1;
    A a3= new A(6);    a1 = a3;
    a2.y= a2.x * 2;    a3.x = a2.y + 1;
    System.out.println(a1.x + ", " + a2.x);
```

EX6. Consider classes A, B and C shown below. Execute the following two statements, showing what is printed.

```
C c = new C();
c.kissme();

public abstract class A {
    public int mamma= 2;
    public abstract void dothis(int x);
    public abstract void kissme();
}

public class B extends A {
    public void dothis(int x) {
        System.out.print("(3)");
        if (x > 0)
            dothis(x);
        kissme();
        kissme();
    }
    public void kissme() {
        System.out.print("(4)");
        if (mamma > 0) {mamma= mamma – 1;  dothis(0) }
        mamma= mamma – 1;
    }
}

public class C extends B {
    public void dothis(int x) {
        System.out.print("(5)");
        while (x > 0) {
            super.dothis(x);
            x= x – 1;
        }
    }
    public void kissme() {
        System.out.print("(6)");
        super.kissme();
        if (mamma > 0) {
            mamma= mamma – 1; super.dothis(1);
        }
    }
}
```

EX7. Exexute the following statement, writing down the strings that appear in the Java console.

```
    Computer c= new Computer(new Q("foo"));

public class Computer {
    public static Q cc= new Q("A");
    public Q mycc= new Q(cc + "B");
    public Computer(Q someQ) {
        cc= new Q(mycc + "Z");
        System.out.println("C" + someQ);
    }
}

public class Q {
    private String name = null;

    public Q(String name) {
        this.name = name;
        System.out.println("Q" + name);
    }

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

QUESTIONS ON LOOPS

In these questions, if a loop invariant is given, it MUST be used in developing the loop. If the complete Java program would require an inner loop, do NOT write the loop; instead, use a mixture of Java an English to write the repetend at a higher level of abstraction. Once you have finished a loop, put it into DrJava and execute it! That will give you assurance that you did it right.

LO1(a). Given a rectangular **int** array a[][] with an even number of rows, write a loop that creates a new array b[][] with its lines being a's lines reflected over its horizontal symmetric axis.

precondition: a.length % 2 == 0
postcondition: Each row b[i] equals row a[a.length–1–i] for each i in 0..a.length–1.
loop invariant: Each row b[i] equals row a[a.length–1–i] for each i in 0..k–1.

LO1(b) Same as LO1(a) but use this invariant:
Loop invariant: Each column b[..][j] equals column a[..][a[0].length–1–j] for each j in 0..k–1.

LO2. The Fibonnaci numbers f_0, f_1, ... are 0, 1, 1, 2, 3, 5, 8, 13, 21, ... Write a loop to calculate F[n] for some n >= 0. Use this invariant:

Invariant: 0 <= i <= n and b = f_i and c = f_{i-1}.

For this purpose, assume that f_{-1} is 1.

LO3. Do the same as LO2, except use this invariant: 0 <= i <= n and b = f_i and c = f_{i+1}

LO4. Write a function with this heading:

```
    // = "x is in array b"
    public static boolean isIn(int[][] b, int x)
```

The invariant should be: x is not in columns b[][0..c–1]
Write the repetend at a high level, stating what it should do; do not write an inner loop.

LO5. Same thing as question 4 but use the invariant: x is not in columns b[][c+1..]

LO6. Same thing as question 4 but use the invariant: x is not in columns b[]c..]

LO7. Your answer from question 4 contains a high-level statement in the repetend. Refine it. It will be a loop with initialization. Write the invariant of the loop first.

LO8. Write a loop to fill in a magic square of odd size. This is a square whose row, columns, and diagonals all lead up to the same thing. Suppose it is an n-by-n square b[][]. Here's how one proceeds. Put the number 1 in some square. Each time you have placed an integer in a square b[i][j], do the following.

Change i and j to move to the square diagonally up and to the right, b[i–1][j–1], and place the next integer there --if that square is already filled, then instead move down to the square b[i+1][j] and put the integer there. Things wrap around, e.g. if i–1 is –1, use n–1 for the row number.

LO9. Write a program segment to zip the two arrays b[] and c[] into a third array d[]. b and c need not be the same size. d should contain b[0], c[0], b[1], c[1], b[2], c[2], ... When one array becomes empty and the other doesn't, just append the rest of the other one. This can be done in one loop, but three may make more sense.

LO10. 1. Write a loop to delete all instances of character 'a' from StringBuffer buf.
invariant: buf[0..i–1] has no 'a'
postcondition: buf[0..buf.length()-1] has no 'a'

LO11.. You're fishing at a lake in New York. You have caught several fish and have stored their weights (in pounds) in a **double** array, weights. According to local fishing laws, you are only allowed to keep fish weighing one pound or more. Given the array, weights, write a loop to calculate the total weight of the fish you can keep.
precondition: weights contains the weights of the fish you caught
invariant: totalWeight is the sum of weights[0..i–1] that you can keep
postcondition: totalWeight is the sum of weights[0..weights.length-1] that you can keep

L012. The Fibonacci sequence is a series of numbers where each number after the first two equals the sum of the previous two. For example, the first 10 numbers of the sequence are: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34. Write a loop to compute the first n numbers of the Fibonacci sequence and store them in array fib.
precondition: n > 1
invariant: fib[0..i–1] contains the first i numbers in the Fibonacci sequence
postcondition: fib[0..n–1] contains the first n numbers in the Fibonacci sequence

LO13. Write a loop to compute the floor of the square root of a nonnegative **int** k and store it in **int** floorOfSqrt. That is, upon completion, floorOfSqrt should be the largest integer that is less than the square root of k. You may not use function Math.sqrt.
precondition: k >= 0
invariant: k < (floorOfSqrt+1)^2
postcondition: floorOfSqrt^2 <= k and k < (floorOfSqrt+1)^2

LO14. Write a loop that implements the indexOf(char) function in class String: find the index of char c in String s.
invariant: indexOfC = s.substring(0,i).indexOf(c).
postcondition: indexOfC = s.indexOf(c).

LO15. Write a loop to determine the number of times a substring sub can be removed from a string s. Use the following postcondition and invariant.
postcondition: n = number of times sub can be removed from s[0..s.length()-1]
invariant: n = number of times sub can be removed from s[0..k-1]

Assume that if k > (s.length() – 1) then s[0..k] means s[0..s.length() –1]
Examples: "abc" can be removed from "abcabc" 2 times
"aa" can be removed from "aaa" 1 time
Hint: For strings a, b and an integer j

a.indexOf(b,j) = index of first letter of the first occurrence of string b in string a at or beyond a[j]. This will be -1 if b does not occur in a at or beyond a[j].

LO16. Write a loop to determine the number of times a substring sub appears in a string s. Use the following postcondition and invariant.
postcondition: n = number of times sub appears in s[0..s.length()–1]
invariant: n = number of times sub appears in s[0..k–1]
Examples: "abc" appears in "abcabc" 2 times
"aa" appears in "aaa" 2 times

"aa" appears in "aaaa" 3 times

Hint: Your solution should be similar to your solution to (1). The difference will be in the way you answer the third loopy question.

LO17. Write a loop to encrypt a message (a String m) by changing each character c in m to the character (**char**)((**int**) c + 13). (Try this in the interactions pane if you'd like to see how this changes the characters.) Some of these characters in the encrypted message will probably not be letters. Use the following invariant and postcondition.

postcondition: em = m[0..m.length()–1] encrypted
invariant: em = m[0..k–1] encrypted

Hint: use String method charAt

LO18. Write a loop to decrypt a message em that has been encoded as in (3). Use the invariant and postcondition below:

postcondition: m = em[0..em.length()–1] decrypted
invariant: m = em[k+1..em.length()–1] decrypted

LO19. Write a loop to replace each integer n in the integer array sums with the integer n*(n+1)/2. Use the following invariant and postcondition.

postcondition: The integers in sums[0..sums.length() – 1] have been 'replaced'
invariant: The integers in sums[0..k–1] have been replaced

LO20. Frodo Baggins has lost his precious ring in the grass, and he needs to get it back before the evil forces find it and destroy the universe. He decided to look for it by walking in a spiral from where he stands: he makes one step ahead, turns right, makes one step ahead again, and turns right again. Next, he makes two steps ahead, turns right, and again, makes two steps again and turns right, and so on, and so on... For increasing values of k, he performs twice the following procedure:

take k steps ahead; turn right

before moving to a higher value of k. He walks until he finds the ring, which is guaranteed to be found.

You are given class frodo, shown below without the actual code, as well as a function to search for the precious ring in a given position, with the following specification:

// = “the ring is at position (x, y)”
public static boolean find_precious(**int** x, **int** y);

Write a single while loop (with all initialization) that will navigate Frodo through the plane by walking along the spiral, and have him search for the ring in every location he visits. Frodo will walk as described above, in *phases*, where in *phase* i Frodo makes two series of i steps ahead, turning right after every series. After finding the precious (find precious returning true), print the location where it was found and the number of steps Frodo had to walk in order to find it. Declare all variables that you are using

Use the following invariant.

p = the number of the phase Frodo is currently in (first one is 1)
s = the number of series of steps Frodo has already completed in this phase (0 or 1)
k = the number of steps made so far in the current series, and k < p
nSteps = the total number of steps Frodo has made along the spiral.
Frodo is standing at a location corresponding to the current values of variables i, j, k, nsteps,
facing in the appropriate direction.
All the preceding locations along the spiral have already been searched

```
public class Frodo {
    // Constructor: a frodo at (0,0) facing north
    public frodo() { ... };

    // = x-coordinate of Frodo’s current position
    public int getX() { ... };
    // = y-coordinate of Frodo's current position
    public int getY() { ... };

    // Make one step in the direction in which Frodo faces
    public void go() { ... };

    // Turn Frodo right w/o moving from the current position
    public void turnRight() { ... };
}
```

LO21. You are analyzing a history of stock prices of a certain stock, stored in array segment stock[0..ndays] of type **double**, where ndays > 0. There is one element for each day. You want to find out what the biggest contiguous price increase of this stock throughout its history (a contiguous price increase is a series of increases of stock value without any decreases of value in the middle). Such a contiguous price increase could be thought of as the best deal a broker can get in short term, buying the stock at the point where the increase starts and selling it immediately at the point where it is about to start going down.

Write a single loop that searches through stock[0..ndays] and finds the biggest such price increase, the day on which it begins, and the day on which it ends. Maintain the following invariant:

```
stock[0..k-1] has been processed, and
startDay is the day on which the most recent price increase in stock[0..k-1] started.
stock[buyingDay..sellingDay] is the biggest contiguous price increase in stock[0..k-1].
BestDeal = stock[sellingDay] – stock[buyingDay]
```

MISCELLANEOUS

MI1. Consider the following piece of code. Write down what appears on the screen after the code is run. It may help you to draw the vector, but you are not required to do so.

```
Vector v= new Vector();
v.add("alpha")        v.add("beta");
v.addElement("gamma");    v.add("delta");
v.add("epsilon");    v.add("alpha");
v.add("eta");          v.add("zeta");
v.add("iota");         v.add("kappa");
v.add("lambda");

v.remove("epsilon");     v.remove("alpha");

System.out.println("index: "+v.indexOf("kappa"));
System.out.println("size: "+v.size());
System.out.println("capacity: "+v.capacity());
```

MI2. What are the types of the following expressions?

```
Integer.parseInt("23");
new Double("23");
```

MI3. Two **int** arrays a and b have been declared and initialized (i.e. they have been filled with integer values). Write Java code to declare a new **int** array c that is exactly large enough to hold all the values in array a and all the values in array b. After you declare c, write Java code to fill the first part of c with the values in a (in the same order that they appear in a) and the last part of c with the values in b. E.g. if

a = {1, 9, -1}   and   b = {2, 0}   then c should be {1, 9, -1, 2, 0}.

If you have to write a loop you may write it any way you like. You do not need to write an invariant.

MI4. (a) Write an expression whose value is the index of the last character of a String s.
(b) What is printed by execution of the following?

```
String s= "Java is fun.";
System.out.println(s.substring(1,6));
```

(c) What is printed out if the following?

```
StringBuffer sb= new StringBuffer("Have a nice break!");
System.out.println(sb.deleteCharAt(2));
```

(d) Write an assignment statement that adds the string "Cornell " to the beginning of String s2.

Answers

OO1.
```
public class SportsCar extends Car{
    private int topSpeed;

    // Constructor: an instance with miles per gallon mpg and top speed topSpeed
    public SportsCar(double mpg, int topSpeed){
        super(mpg);
        this.topSpeed = topSpeed;
    }

    /* = A string that gives the fuel economy and top speed of
    this sports car.  Hint:  This string will be more than one sentence.  */
    public String toString(){
       return super.toString() + "  This car has a top speed of " + topSpeed + ".";
    }

    // = the top speed of this car
    public int getTopSpeed(){
        return topSpeed;
    }
}
```

OO2. (a) Look at your textbook if you don't know how to do this.

(b) Variable **'this'** would have the name of the folder created in (a) –(a0 is a common name we have used in lecture)

(c) False, private variables cannot be referenced directly in a subclass.

OO3. SportsCar sc= **new** SportsCar(15.0, 170);   Legal
Car mySportsCar= sc;   Legal
int ts= mySportsCar.getTopSpeed(); Illegal (apparent class of mySportsCar is Car)
double ts2= sc.getTopSpeed();   Legal (you can assign an **int** value to a **double** variable; the **int** gets promoted to a **double**.)
Car c= **new** Car(true, "Red");   Illegal, the only constructor takes a double.
Car myOtherCar= **new** Car(45.2); Legal
SportsCar sc2**=** myOtherCar;   Illegal

OO4. (a) Since Book is not abstract, it must override the method getPrintedLanguage. That is, Book must have a method getPrintedLanguage.

(b)
```
public Book(int numPages, String language, String title, String author){
     this.numPages= numPages;
     printedLanguage= language;
     this.title= title;
     this.author= author;
    }
```

(c) Book b= **new** Book(301, "English", "The Brothers Karamazov", "Dostoyevsky"); Legal
PrintedMaterial pm= b;   Legal
PrintedMaterial pm2= **new** PrintedMaterial(); Illegal (you can't instantiate an abstract class)

PrintedMaterial[] pmArray= **new** PrintedMaterial[9]; Legal

OO5. (a) putOnSale(.50);
(b)
```
public Shirt(double price, String manufacturer, String size)
        { super(price, manufacturer); this.size = size; }
    public String getSize() { return size; }
    public String toString() {
        return "Shirt[price=" + getPrice() + ", manufacturer=" +
                    getManufacturer() + ", size=" + size + "]";
    }
```
(c) Shirt, Shirt, RetailItem, Shirt, Yes, RetailItem does not have a getSize method defined, No, Shirt[price=80.5, manufacturer=Ralph Lauren, size=M], (or whatever toString class Shirt class returns).

```
/** An instance represents a Customer */
public class Customer {
    private static int nextIDNumber = 1; // The next id number to be used
    private int idNumber; // customer ID number
    private String name; // customer name
    private ShoppingCart items; // customer's shopping cart
    private Address mailingAddress; // customer's mailing address

    /** Constructor a new customer object with idNumber, name, and mailing address */
    public Customer(String name, Address mailingAddress) {
        this.idNumber = nextIDNumber;
        nextIDNumber= nextIDNumber + 1;
        this.name = name;
        this.mailingAddress = mailingAddress;
        this.items = new ShoppingCart();
    }
}
```

OO7. (a) 4, 6, 13, (b) 9, (c) true, false, no response, tears, true, NC-17.

OO8 (a) none, (b) getFrequency(), toInt(), (c) equals(Object), toString(), any method in Object.

OO9. class Vehicle 1. model== mod; is not a statement. should be model= mod;.
2. getNumWheels() is missing a return statement.
3. return model is missing a semicolon.
4. in method main, you can't create a new Vehicle because it's an abstract class.

class Car: 1. the assignment numDoors= numDoors; assigns to the local variable, not the class field. should be **this**.numDoors= numDoors;.
2. the call to the super constructor has to be the first line of Car's constructor.
3. in method main, you can't create c2 because the default constructor Car() is not defined.

OO10. (a) **this**.population= population; (b) **super**(222000000); (c) **return** population;
(c) population= population + increase;. to decrease, maker the argument of changePopulation(**long**) negative.
(e) System.out.println("USA's population: "+ getPopulation());

OO11.
1. Abstract method cannot have a body; method Boy.sayHello() should not be declared abstract.
2. Cannot refer to private field myname in class GoodBoy. Either change the field to public or add a getter method getName() to Boy and use it in GoodBoy.
3. Cannot construct an object of abstract type (there is no way to fix it, it is just fundamentally wrong).

EX1 You will draw four frames, with method name "isPalindrome" and scopebox "Word"
First frame parameters: word: racecar, start: 0, end: 6
Second frame parameters: word: racecar, start: 1, end: 5
Third frame parameters: word: racecar, start: 2, end: 4
Fourth frame parameters: word: racecar, start: 3, end: 3
All returned values are true.

EX2. Jesse
Jesse
Blah
Jesse
Bar
False

EX3. sqrt
int
cube
constructor
2

EX4. 110

EX5. 81, 40

EX6. (6)(4)(5)

EX7. Qfoo
QA
QAB
QABZ
Cfoo

LO10.
```
int i= 0;
   // inv: buf[0..i–1] has no 'a'
   while (i != buf.length()) {
            if (buf.charAt(i) == 'a') buf.delete(i, i+1);
            else i= i+1;
   }
```

LO11.
```
// pre: weights contains the weights of the fish you caught
      double totalWeight= 0;
      int i= 0;
      // inv: totalWeight is the sum of weights[0..i–1] that you can keep
      while (i != weights.length) {
             if (weights[i] >= 1.0)
                  totalWeight= totalWeight + weights[i];
            i= i+1;
      }
      // post: totalWeight is the sum of weights[0..weights.length-1] that you can keep
```

LO12.
```
// precondition: n > 1
      int[] fib= new int[n];
      fib[0]= 0;     fib[1]= 1;     int i= 2;
       // inv: fib[0..i–1] contains the first i numbers in the Fibonacci sequence
      while (i != n) {  fib[i]= fib[i–2] + fib[i–1];  i= i+1; }
      // post: fib[0..n-1] contains the first n numbers in the Fibonacci sequence
```

LO13.
```
// precondition: k >= 0
      int floorOfSqrt= k;
      // invariant: k < (floorOfSqrt+1)^2
      while (k > floorOfSqrt*floorOfSqrt)
            floorOfSqrt= floorOfSqrt – 1;
      // postcondition: floorOfSqrt^2 <= k  and k < (floorOfSqrt+1)^2
```

LO14
```
int indexOfC= -1;
      //invariant: indexOfC =  s.substring(0,i).indexOf(c)
```

```
    for (int i= 0; i != s.length() && indexOfC == –1; i= i+1) {
        if (s.charAt(i) == c) indexOfC= i;
    }
  // postcondition: indexOfC = s.indexOf(c)
```

```
LO15. int k= 0;   int n= 0;
    //invariant:  n = number of times sub can be removed from s[0..k–1]
    while (k < s.length() – 1){
        int i= s.indexOf(sub,k);
        if (i > –1){ //sub appears at or beyond index k
            n= n + 1;  //count that occurrence
            k= i + sub.length();
        }
        else{ //sub does not appear in the rest of the string
            k = s.length() – 1;
        }
    }
    //postcondition:  n = number of times sub can be removed from s[0..s.length()–1]
```

```
 LO16. int k = 0;   int n = 0;
    //invariant:     n = number of times sub appears in s[0..k–1]
    while (k < s.length() – 1) {
        int i= s.indexOf(sub,k);
        if (i != –1){ // sub appears at or beyond index k
            n= n + 1;  //count that occurrence
            k= i + 1;
        }
        else {  //sub does not appear in the rest of the string
            k= s.length() – 1;
        }
   }
    //postcondition:  n = number of times sub appears in s[0..s.length()–1]
```

```
LO17.    int k = 0;  String em = "";
    //invariant:  em = m[0..k–1] encrypted
    while ( k!= m.length()) {
        char c= m.charAt(k);
        em= em + (char)((int) c + 13);
        k= k + 1;
    }
    //postcondition:  em = m[0..m.length()–1] encrypted
```

```
LO18.    int k= em.length() - 1;
    String m= "";
    //invariant:     m = em[k+1..em.length()–1] decrypted
    while (k != –1) {
        char c= em.charAt(k);
        m= (char)((int) c – 13) + m;
        k= k – 1;
    }
    //postcondition:  m = em[0..em.length()-1] decrypted
```

```
LO19.  int k= 0;
    //invariant: The integers in sums[0..k–1] have been replaced
    while (k != sums.length()) {
        sums[k]= sums[k]*(sums[k]+1)/2;
        k= k + 1;
    }
    //postcondition:  The integers in sums[0..sums.length() – 1] have been 'replaced'
```

```
LO20. frodo f= new frodo();
```

```
int nSteps= 0;
int p= 1;
int s= 0;
int k= 0;
while (!find_precious(f.getX(), f.getY())) {
    // Take a step forward in the current series
        f.go(); k= k+1; nSteps= nSteps + 1;

    if (k == p) {
        // A series of steps has been completed. Turn right and move to next series or phase
        f.turnRight(); s= s + 1; k= 0;
        if (s == 2) {
            // Frodo has completed 2 series. Move to next phase
            p= p+1; s= 0; k= 0;
        }
    }
}
System.out.println("Found precious at: " + f.getX() + ", " + f.getY() + " after " + nsteps + " steps.");
```

LO21.
```
int startDay= 0;
int buyingDay= 0;
int sellingDay= 0;
double bestDeal= 0;
// invariant: as in question
for (int k= 1; k != ndays; k= k+1) {
    if (stock[k-1] <= stock[k]) {
        // { stock[startDay..k] is nondecreasing }
        double thisDeal= stock[k] - stock[startday];
        if (thisDeal > bestDeal)      {
            bestDeal= thisDeal;
            buyingDay= startDay;
            sellingDay= k;
        }
    }
    else { startDay= k; }
}
```

MI1. index: 2
size: 9
capacity: 20

MI2. **int**, Double

M13.
```
int[] c= new int[a.length + b.length];
    //put a's values into c
        for (int i= 0; i < a.length; i= i+1){
            c[i]= a[i];
        }
    //put b's values into c
        for (int j= a.length; j < c.length; j= j+1){
            c[j] = b[j – a.length];
        }
```

M14. (a) s.length() – 1
(b) ava i
(c) Hae a nice break!
(d) s2= "Cornell " + s2;