CSCI 1101
Computer Science II
PRACTICE SET FOR TEST NO.1
SOLUTIONS

1 . For each of the following questions, select the most appropriate answer.

1. A class in java is like
   a. a variable
   b. an object
   c. an instantiation of an object
   d. a blueprint for creating objects

2. This key word causes an object to be created in memory
   a. create
   b. new
   c. instantiate
   d. construct

3. The signature of a method consists of the following
   a. Return type, method name and types of input parameters
   b. Method name and types of input parameters
   c. Method name, types of input parameters and their order
   d. Return type, method name, types of input parameters and their order

4. A static method
   a. Can be called by only one object
   b. Can be called without object instantiation
   c. Has to be called from inside the class
   d. Cannot be called by non-static methods

5. The following is true with respect to a static variable
   a. It can be modified by only one object
   b. Each object gets a unique copy of the static variable
   c. Only one copy exists – no matter how many objects are instantiated
   d. Cannot be modified by non-static method
6. The get and set methods in a class are also known, respectively, as
   a. accessor and mutator methods
   b. mutator and accessor methods
   c. accessor and constructor methods
   d. mutator and constructor methods.
7. If a class is named Student, what name can you use for a constructor for this class?
   a. Student
   b. Student.java
   c. void Student
   d. any name can be used since the constructor can be overloaded
8. When defining a constructor, what do you specify for the type of value returned?
   a. Primitive data type such as int
   b. Class name such as Student
   c. void
   d. **None – a constructor does not have an explicit return type**

9. The following method defined in a class called MyClass has the header

```
public void myMethod(int n)
```

Which of the following is a correct call to the method from outside MyClass. Assume that myObj is an object of type MyClass. Choose all correct answers.

a. **`myObj.myMethod(10);`**
b. `myObj.MyClass.myMethod(10);`
c. `MyClass.myMethod(10);`
d. `MyClass(10);`

10. The advantage of a deep copy in a constructor is the following:
a. deep copy saves memory
b. **deep copy provides better encapsulation**
c. deep copy enables the constructor to be used in multiple classes
d. none of the above – you cannot create a deep copy

11. The following are two valid statements in a demo program that uses a class called MyClass. Assume that object1 is an object of type MyClass and method1 and method2 are two methods in MyClass.

object1.method1(10, "Joe");
MyClass.method2(4.156);

Which of the following is correct (select all correct answers)?
a. method1 and method2 are non-static methods
b. method1 is a non-static method and method2 is a static method
c. method1 is a static method and method2 is a non-static method
d. method2 is a static method while method1 could be static or non-static

12. Aggregation can be identified by
a. An "is-a" relationship between objects
b. A "has-a" relationship between objects
c. No relationship between objects
d. A "may be a" relationship between objects

13. This keyword indicates that a class inherits from another class
a. derived
b. super
c. **extends**
d. this

14. Which constructor would create an instance of class Person that has two private attributes: String name and int age?
a. public Person(String n, int a) { n = name; a = age;}
b. public void Person(String n, int a) {name = n; age = a;}
c. public void Person(String name, int age){this.name= name; this.age=age;}
d. public Person(String name, int age){this.name=name; this.age=age;}

15. When you implement a method with the same name as another method inside a class, you _____ the original method.

a. **overload**
b. override
c. copy
d. aggregate

16. Analyze the following code:

```
public class SomeClass {
      private double info;

       public SomeClass(double info) {
            info = info;
      }
}
```

a. The program has a compilation error because you cannot assign info to info.
b. The program does not compile because SomeClass does not have a default constructor.
c. **The program will compile, but you cannot create an object of SomeClass with a specified info. The object will always have info set to 0.**
d. The program has a compilation error because it does not have a main method.

17. Analyze the following code.

```
public class Test {
      public void xMethod(int n){
            n++;
      }
}
```

Another class with a main method uses the above Test class as follows:

```
      public static void main(String[] args){
            int n = 2;
            System.out.println(Test.xMethod(n));
      }
}
```

a. The code has a syntax error because xMethod does not return a value.
b. The code prints 3.
c. **The code has a syntax error because xMethod is not declared static**.
d. The code prints 2.
e. The code prints 0.

18. Class A inherits from Class B, Class C is the superclass for Class B. Which of the following group of class headers is correct (complete code omitted)?

a. public class B extends A {} and public class C extends B {}
b. public class B extends A {} and public class B extends C {}
c. **public class A extends B {} and public class B extends C {}**
d. public class A extends B {} and public class C extends B {}

QUESTION NO. 2 Longer Multiple Choice Questions

For questions 1 to 6 below, please refer to the Dog class given below. Some statements are missing and by answering the multiple-choice questions, you will build the missing pieces. Here's a general description of the Dog class.

Dog has two attributes: name (String) and age (int). The constructor sets both attributes. There are get and set methods. In addition the class has the following:

- A method called speak that returns a String "bark"
- Another method called speak that just returns a given String
- A method isOlder that returns true if "this" Dog object is older than another Dog object
- A method isSame that compares this Dog object with another and returns true if the name and age are the same
- A method copy that returns a copy of this Dog object with the same name and age.

```
1.  public class Dog {
2.  //attributes
3.  private String name;
4.  private int age;
5.  //Constructor
6.  //this line is still missing
7.  {
            this.name=name;
            this.age=age;
8.  }
9.  //get and set methods
10. public String getName() {return name;}
11. public int getAge() {return age;}
12. public void setName(String n){name=n;}
13. public void setAge (int a) {age=a;}
14. //speak method
15. public String speak (){
16. return "bark";
17. }
18. //the second speak method
19. //this line is still missing
20. return s; }

21. //isOlder method
22. public boolean isOlder (Dog d){
23. //this line is still missing
24. }

25. //isSame method
26. public boolean isSame (Dog d) {
27. //this line is still missing
28. }

29. //copy method
30. //missing code
31. //missing code

32. public String toString(){
33. return name + " is " + age + " years old";}
```

1. Line 6 (choose the correct code to fill in the missing piece in line no. 6)
   a. `public Dog(String n, String a)`
   b. `public void Dog(String name, int age)`
   c. `public void Dog(String n, int a)`
   d. `public Dog(String name, int age)`
2. Line 19
   a. `public void speak(String bark){`
   b. `public String speak(String bark){`
   c. `public void speak(String s){`
   d. `public String speak(String s){`
3. Line 23
   a. `return this.dog>d.getDog();`
   b. `return this.age>d.getAge();`
   c. `return age>other.age;`
   d. `return this.getAge()>other.age;`
4. Line 27
   a. `return (name.equals(d.getName())&&age==d.getAge());`
   b. `return this==d;`
   c. `return this.name==d.name&&this.age==d.age;`
   d. `return this.name.equals(d.getName())&&this.age==d.age;`

5. Lines 30 and 31

a. public void copy(){
   return new Dog(name, age);}

b. public Dog copy(){
**   return new Dog(name, age);}**

c. public Dog copy(Dog d){
   return new Dog();}

d. public void copy(Dog d){
   return new Dog();}

6. Look at the three demo programs given below. What demo program will produce the following output?

```
 ----jGRASP exec: java DogDemo
Zara is 2 years old
Happy is 1 years old

Zara is older than Happy
Woof

 ----jGRASP: operation complete.
```

Select the correct answer here:

a. Demo program 1 only
b. Demo program 2 only
c. Demo program 3 only
d. Demo programs 1 and 2
e. Demo programs 1 and 3

```
//Demo program 1
public class DogDemo {
  public static void main (String [] args){
    Dog husky = new Dog (2, "Zara");
    Dog lab = new Dog (1, "Happy");
    System.out.println(husky);
    System.out.println(lab);
    System.out.println();
    if (husky.isOlder(lab))
      System.out.println(husky.getName() + " is older than " + lab.getName());
    else
      System.out.println(lab.getName()+ " is older than " + husky.getName());
   System.out.println(husky.speak());
  }
}
```

```
//Demo program 2
public class DogDemo {
  public static void main (String [] args){
    Dog husky = new Dog ("Zara", 2);
    Dog lab = new Dog ("Happy", 1);
    System.out.println(husky);
    System.out.println(lab);
    System.out.println();
    if (husky.isOlder(lab))
      System.out.println(husky.getName() + " is older
      than " + lab.getName());
    else
      System.out.println(lab.getName()+ " is older
      than " + husky.getName());
   System.out.println(husky.speak("Woof"));
   System.out.println(husky.speak());
  }
}
```

```
//Demo program 3
public class DogDemo {
  public static void main (String [] args){
    Dog husky = new Dog ("Zara", 2);
    Dog lab = new Dog ("Happy", 1);
    System.out.println(husky);
    System.out.println(lab);
    System.out.println();
    if (husky.isOlder(lab))
      System.out.println(husky.getName() + " is older
       than " + lab.getName());
    else
      System.out.println(lab.getName()+ " is older
      than " + husky.getName());
   System.out.println(husky.speak("Woof"));
  }
}
```

For questions 7 to 15 below, you will develop two simple classes, namely, Pokemon.java and PokemonDeck.java and answer questions about it.

Consider a class to represent a Pokemon card. It has partial code given below:

```
//This class represents a Pokemon card. The card has a name
//(String) and hp (an int that represents healthpoints, that is,
//it indicates how much damage this Pokemon can cause before
//fainting). There is also a static variable price that
//represents the price of Pokemon cards.
public class Pokemon
{
     private String name;
     private int hp;
     private static double price;

     //rest of the code missing
}
```

7. Suppose you wanted to write a method in the Pokemon class that would allow another class to access the hp of this Pokemon. What would the header of such a method look like?
   a. `public static int getHP()`
   b. `public int getHP()`
   c. `public static getHP(int hp)`
   d. `public int getHP(int hp)`

8. Suppose you wanted to write a method in the Pokemon class that compares the hp of "this" Pokemon with the hp of another Pokemon and returns true if the hp of "this" Pokemon is greater. What would the header for such a method look like?
   a. `public boolean compareHP(this.hp)`
   b. `public void compareHP(other.getHP())`
   c. `public void compareHP(Pokemon other)`
   d. `public boolean compareHP(Pokemon other)`

9. Suppose you wanted to write a method in the Pokemon class to set the price to a given value. What would the header for such a class look like?
   a. `public double setPrice(double p)`
   b. `public void setPrice()`
   c. `public static double setPrice(double p)`
   d. `public static void setPrice(double p)`

Now suppose you want to write another class called PokemonDeck that uses or "is aggregated from" the Pokemon class. Its attributes are an array of Pokemon card objects, a capacity (how many Pokemon card objects that the array can hold), the current number of Pokemon objects in the array, and the total hp of all the Pokemon in the array.

```
public class PokemonDeck
{
      //missing declaration (an array of Pokemon card objects)

      private int capacity;       //size of the array

      private int num;            //current number of Pokemon card
                                  //objects in the array

      private int totalHP;  //total HP of all the Pokemon in
                                  //the array

      //rest of the code missing
}
```

10. What would you write in the missing declaration above?
    a. private int[] deck;
    b. private Pokemon[] deck;
    c. private String[] deck;
    d. private static String[] deck;
11. Suppose you wanted to add a method to the PokemonDeck class called addPokemon that adds a Pokemon object to the array. What would be the header of such a method?

    a. public String addPokemon()
    b. public Pokemon[] addPokemon(Pokemon p)
    c. public void addPokemon(Pokemon p)
    d. public String addPokemon(Pokemon p)
12. Continuing from Question 11, which of the following would be the correct set of statements that you would have in the method addPokemon?

```
a. if (num<capacity){
        Pokemon[num] = p;
        num++;}
b. if (num>capacity){
        Pokemon[num]=p;
        num++;}

c. if (num<capacity){
        deck[num]=p;
        num++;}
d. if (num>capacity){
        deck[num]=p;
        num++;}
```

13. Suppose you wanted to display the name of the first Pokemon in the deck (assume that the array has at least one Pokemon in it). What would be the statement?

```
a. System.out.println(Pokemon[0].getName());
b. System.out.println(deck[0].getName());
c. System.out.println(deck[1].name);
d. System.out.println(deck.getName[0]());
```

14. Suppose you wanted to write a method in the PokemonDeck class to calculate the totalHP of all the Pokemon in the array. What would be the header of such a method?

```
a. public static void findTotalHP()
b. public void findTotalHP()
c. public static int findTotalHP()
d. public int findTotalHP()
```

15. Continuing the question 14 above, what would be the for loop inside that method?

```
a. for(int i=0; i<num; i++){totalHP+=deck[i].getHP();}

b. for(int i=0; i<=num; i++){totalHP+=deck[i].getHP(hp);}

c. for(int i=0; i<=num;i++){totalHP+=Pokemon[i].getHP();}

d. for(int i=0; i<num; i++)
        {totalHP+=Pokemon[i].getHP(hp);}
```

3.Write the signatures of each of the following pairs of method headers. Also write if the pair can exist in the same class.

- public String add(String str1, String str2)
  add(String, String)

  public String add(int num1, int num2)
  add(int,int) → can be overloaded

- public int add(String str, int num)
  add(String, int)

  public int add(int num, String str)
  add(int, String) → can be overloaded

- public void add(ClassA a, ClassB b)
  add(ClassA, ClassB)
  overloaded

  public void add(ClassA b, ClassB a)
  add(ClassA, ClassB) → cannot be

4. What is the error in the following class definition?

```
public class MyClass
{
        private int x;
        private double y;
        public static void setValues(int a, double b)
        {
                x = a;
                y = b;
        }
}
```

Error: Static method trying to modify instance variables

5. Assume that you have the following class definition.

```
public class MyClass
{
      private String name;

      public void setName(String name)
      {
            this.name = name;
      }
}
```

Suppose a main method has the following statements. Which of the following are legal (that is, will compile and run), giving reasons as necessary.

a) `MyClass joe = new MyClass("Joe");` → **illegal; no constructor with String as input parameter**

b) `MyClass joe = new MyClass();` → **legal (default constructor)**
`joe.name = "Joe";` → **illegal (name is declared private)**

c) `MyClass joe = new MyClass();` → **legal**
`joe.setName("Joe");` → **legal**

d) `MyClass joe = new MyClass();` → **legal**
`System.out.println(joe);` → **will compile and run, but will display memory value since there is no toString method.**

6. A vehicle has a registration number, maximum capacity of the gas tank, amount of gas in the tank and gas consumption rate (number of liters of gas consumed per km traveled). When a vehicle object is instantiated, its gas tank is set to empty.

a. Write a class called Vehicle that includes the above attributes, a constructor method that sets the registration number, gas consumption rate and capacity, a method to pump the specified amount of gas into the tank, a method to get the amount of gas in the tank, and a method travel that makes the car travel for the specified distance.

b. Write a test program that creates a car with registration number BLA856, gas consumption rate of 1 liter/12 km, and gas tank with capacity of 60 liters. In the test program, include a line of java to:
   i. Pump 20 liters of gas into the tank
   ii. Make the car travel 45 km.
   iii. Print the amount of gas remaining in the tank

```
//Vehicle.java
public class Vehicle{
	private String regNo;
	private double capacity;
	private double amount;
	private double rate;

	public Vehicle(String reg, double c, double r){
		regNo = reg;
		capacity = c;
		rate = r;
		amount = 0;
	}
	public void pump(double a){
		if (amount+a > capacity)
		{
			System.out.println("Can't pump. Exceeds capacity");
		}
		else
			amount = amount+a;
	}
	public double getGas(){
		return amount;
	}
	public void travel(int dist){
		double temp = amount - dist*rate;
		if (temp<0)

			System.out.println("Cannot travel. Insufficient gas");
		else
			amount = temp;
	}
}

//VehicleDemo.java
public class VehicleDemo{
	public static void main(String[] args){
		Vehicle myCar = new Vehicle("BLA856", 60, 0.08);
		myCar.pump(20);
		myCar.travel(45);
		System.out.println("Gas in tank: " + myCar.getGas());
	}
}
```

7. What is the output of the DemoDemo program?

```
public class Demo{
            private int x;
            private static int y = 5;
            public Demo(int value)(
                    x = value;
             }
            public void method1(){
                    x++;
                    y++;
             }
             public int method2(){
                     return x+y;
              }
}

public class DemoDemo{
            public static void main(String[] args){
                     for(int i = 0; i<4; i++){
                              Demo obj1 = new Demo(10);
                              obj1.method1();
                              System.out.println(obj1.method2());

                       }
              }
}
```

Output:
17
18
19
20

8. Create a class RoomOccupancy that can be used to record the number of people in rooms of a hotel. The class has the attributes

- num– the number of people in a room
- max – the maximum number of people allowed in the room
- total – the total number of people in all rooms (a static variable)

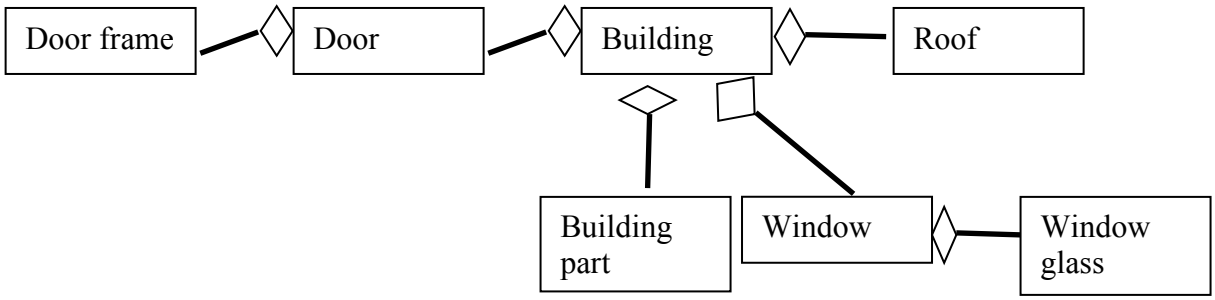
and the following methods

- Constructor that creates a room with number of people =0 and sets max to a given value
- addOneToRoom – adds a person to the room (non-static method)
- removeOneFromRoom – removes a person from the room (non-static method)
- getNumber – gets the number of people in the room (non-static method)
- getTotal – gets the total number of people in all rooms (static method)
- isMore – determines if the number of occupants in this room is greater than that in another room (non-static method)

```
public class RoomOccupancy
{
	private int num;
	private int max;
	private static int total=0;

	public RoomOccupancy(int m)
	{
		max = m;
		num=0;
	}
	public void addOneToRoom()
	{
		if (num<max)
		{
			num++;
			total++;
		}
		else
			System.out.println("Cannot add. Exceeds capacity");
	}
	public void removeOneFromRoom()
	{
		if (num>0)
		{
			num--;
			total--;
		}
		else
			System.out.println("Cannot remove. Has no one!");
	}
	public int getNum()
	{
		return num;
	}
	public static int getTotal()
	{
		return total;
	}
	public boolean isMore(RoomOccupancy r)
	{
		if (num > r.getNum ())
			return true;
		else
			return false;
	}
}
```

9.List all the “has-a” relationships that exist between the following classes using UML notation:

Building, Building part, Roof, Door, Window, Window glass, Door frame



10. For each of the following groups of classes, state whether it is a case of inheritance, or aggregation, or both, or neither. You need to show the relationship ***between every pair*** in each question.

- RetailStore , Manager and Walmart
- Course, Student, CS1101, Instructor, Faculty member

Retail Store – Manager: Aggregation
Walmart - Manager: Aggregation
Walmart – RetailStore: Inheritance

Use similar logic for the second grouping.

11. Determine the output of the following program:

```
public class Check1
{
    private int x;
    private static int y = 10;
    private static int z = 2;
    public Check1(int value)
    {
       x = value;
    }
    public void method1()
    {
       x++;
       y++;
       z--;
    }
    public int method2()
    {
       return x+y+z;
    }
    public static void main(String[] args)
    {
       for(int i = 0; i<2; i++)
       {
           Check1 obj1 = new Check1(10);
           obj1.method1();
           System.out.println(obj1.method2());
       }
    }
}
```

Output:

23
23

12. Consider the following code segment. Use it to answer the questions given below.

```
int n1, n2 = 0;
double x = 12.34;
Turtle t1 = new Turtle();
World w1 = new World(t1);
Turtle t2 = t1;
```

1. How many variables in total are declared in this code segment?
   (a) 0
   (b) 5
   (c) 6
   (d) 7
   (e) none of the above

2. How many objects are created in this code segment?
   (a) 0
   (b) 1
   (c) 2
   (d) 3
   (e) none of the above

3. Write the header for the constructor for the Turtle class:

public Turtle()

4. Write the header for the constructor for the World class

public World(Turtle t)

5. Which of the following statements is true?

   (a) The World class is aggregated from the Turtle class
   (b) The Turtle class is aggregated from the World class

6. Write the UML diagram showing the relationship between the Turtle class and the World class.

Draw the diagram showing an arrow with a diamond tip from Turtle to World.

13. Consider a StockPurchase class

- The Stock symbol and its price
- Number of shares
- Total value

The class is aggregated from a Stock class that holds the Stock symbol and its price.

Write the following:

- Attributes for Stock class
- Constructor for Stock class (that sets the symbol and price)
- Attributes for StockPurchase class
- Deep copy constructor for StockPurchase class (that sets the Stock, number of shares and finds the total value)
- Shallow copy constructor for StockPurchase class (that sets the Stock, number of shares and finds the total value)

Attributes for Stock:

```
private String symbol;
private double price;
```

Constructor for Stock:

```
public Stock(String sym, double p)
{
        symbol = sym;
        price = p;
}
```

Attributes for StockPurchase:

```
private Stock stock;
private int shares;
private double value;
```

Shallow copy:

```
public StockPurcahse(Stock s, int num)
{
        stock = s;
        shares = num;
        value = num*s.getPrice();
}
```

Deep copy:

```
public StockPurchase(Stock s, int num)
{
        stock = new Stock(s.getSymbol(), s.getPrice());
        shares = num;
        value = num*s.getPrice();
}
```

14. Consider the following class:

```
public class Book
{
    private int pages;
    public Book(){
    }
    public void setPages(int n){
        pages = n;
    }
    public int getPages(){
        return pages;
    }
}
```

Create a class Dictionary that extends the Book class. The main test method for Dictionary and the corresponding output are given. Fill in the rest of the code.

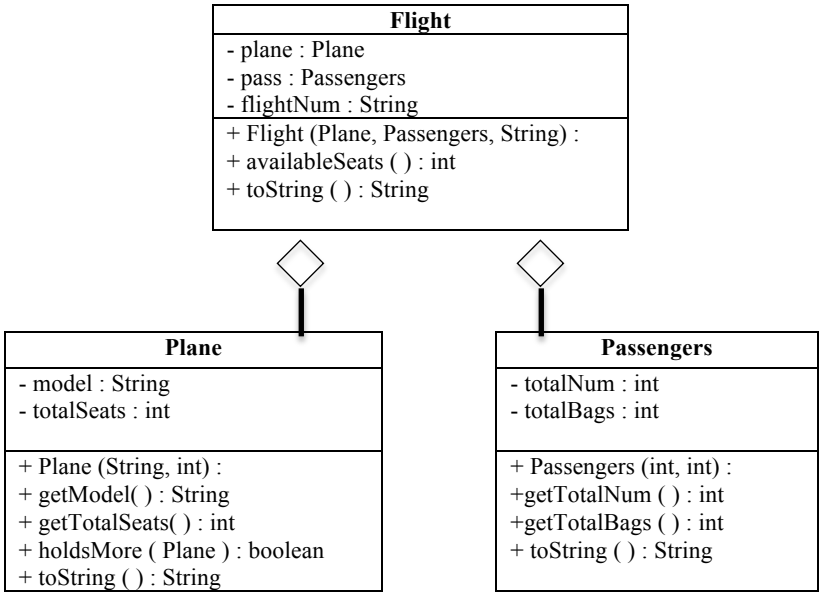
```
public class Dictionary extends Book
{
    private int definitions;
    private String from;
    private String to;
    //fill in code here
    public Dictionary(String from, String to, int pages, int definitions)
    {
        this.from = from;
        this.to = to;
        setPages(pages);
        this.definitions(definitions);
    }
    public String toString()
    {
        String result = from + " to " + to + " Dictionary\n";
        result += ("Number of pages: " + getPages() + "\n");
        result += ("Number of definitions: "  + definitions + "\n");
        return result;
    }
}

public class DictionaryDemo{
    public static void main(String[] args)
    {
        Dictionary webster = new Dictionary("English", "English", 1000, 52500);
        System.out.println(webster);
    }
}
```

The output is:

English to English Dictionary
Number of pages: 1000
Number of definitions: 52500

15. A Flight class that tracks available seats on a given flight has been aggregated from the Plane class (that has the model type and total number of seats) and a Passengers class (that has the total number of passengers booked on a flight and total number of bags (luggage) that will fly together on the flight).

The UML diagram for the three classes is given below. Study the diagram and answer the questions below.



a. Write the constructor for Flight. In the constructor, write a deep copy of Plane and shallow copy of Passengers.

```
public Flight (Plane p, Passengers ps, String fl)
{
   flightNum=fl;
   plane = new Plane (p.getModel(), p.getTotalSeats()); //deep copy
   passenger=ps; //shallow copy
 }
```

b. Write the method holdsMore in the Plane class (compares which plane has more number of total seats).

```
public boolean holdsMore (Plane b)
{
   return ( totalSeats>(b.getTotalSeats()) );
}
```

c. Write the availableSeats in the Flight class (how many more empty seats are available on the flight, return 0 (zero) if the flight is full or if the flight is overbooked i.e., there are more passengers than seats on the plane)

```
public int availableSeats()
{
  int seats = plane.getTotalSeats()-passenger.getTotalNum();
  if (seats<0) //overbooked - more passengers than seats
    return 0;
  else
    return seats;
}
```

d. See the sample output for the Flight class toString method in the FlightDemo class below. Now write the toString methods for the Plane class, Passengers class and the Flight class. Make sure that the Flight class's toString method uses the toString methods of the Plane class and the Passengers class.

```
//Flight toString
public String toString() {
    return "Flight Num: " + flightNum +  " has " + availableSeats() + " available seats" + "\n"+plane+"\n"+passenger;
}

//Passenger toString
public String toString() {
    return ("Total Passengers: " + totalNum + "  Total Bags: " + totalBags);
  }

//Plane toString
public String toString(){
    return "Model: " + model+ "  Total Seats: " + totalSeats;
}
```

```
public class FlightDemo {
   public static void main (String args[]) {
      Plane boeing = new Plane ("Boeing747", 200);
      Passengers pass = new Passengers (150, 60);
      Flight flight = new Flight (boeing, pass, "AC_123");
      System.out.println(flight);
   }
}

//Output
Flight Num: AC_123 has 50 available seats
Model: Boeing747  Total Seats: 200
Total Passengers: 150  Total Bags: 60
```

16. Consider the following class definition:

```
public class Shoe
{
  private double size;
  public Shoe(){  }
  public double getSize(){return size;}
  public void setSize (double s) {size=s;}

  public String toString()
  {
  return ("Shoe Size: " + size);
  }
}
```

Write a class called RunningShoe that extends the Shoe class. The RunningShoe class has colour as its instance variable, a constructor that sets the size and the colour, appropriate get and set methods, and a toString method that prints the size and the colour of the RunningShoe.

```
public class RunningShoe extends Shoe {
  private String colour;
  public RunningShoe (String colour, double size){
    setSize(size);
    this.colour=colour;
  }
  public void setColour (String c) {
    colour=c;
 }
  public String getColour() {
    return colour;
  }
 public String toString (){
    return "Size: " + getSize() + " Colour: " + colour;
 }
}
```