Practice Problems for Exam 1 – Inheritance

Problem 1. Short answer problems.

- The ability to associate many different behaviors to one method name, making the method execute differently depending on in what context the method is called __________

- Briefly describe the difference between abstract and concrete classes.

- Describe the difference between overriding and overloading.

- Which class is at the root of the inheritance hierarchy in Java?

- Which describes a more general type: superclass or subclass?

- Operator __________ can be used to test if an object is an instance of a given class.
Problem 2. Short answer problems: inheritance, polymorphism.
Given the following definitions, answer questions 1-3

```java
public interface OnSale {
    double askingPrice(); // returns the asking price for an item
}
```

```java
public abstract class HotelAccomodation implements OnSale {
    protected final static double STANDARD_PRICE = 120;
    public final static String NONE = "No bed";
    public final static String KING = "1 King";
    public final static String TWO_DOUBLES = "2 Doubles";
    private String number;
    private double price;
    public HotelAccomodation() {
        this.price = STANDARD_PRICE;
        this.number = "000";
    }
    public HotelAccomodation(String num, double price) {
        this.price = price;
        this.number = num;
    }
    public double getPrice() {return this.price;}
    public void setPrice(double newprice) {this.price = newprice;}
    public String getNumber() {return this.number;}
    public String toString() {
        return "Accomodation #" + this.number;
    }
    abstract public String bedType();
    public double askingPrice() {
        return this.price*100;
    }
}
```

```java
public class GuestRoom extends HotelAccomodation {
    private String bedType;
    public GuestRoom (){
        this.bedType = HotelAccomodation.TWO_DOUBLES;
    }
    public GuestRoom (String number, double price, String bed){
        super(number, price);
        this.bedType = bed;
    }
    public String toString (){
        return "Guest Room #" + this.getNumber();
    }
}
```
public class Suite extends HotelAccommodation {
    private int nRooms;
    
    public Suite(String num, double price, int rooms) {
        super(num, price);
        this.nRooms = rooms;
    }
    
    public int getNRooms() { return this.nRooms; }
    
    public String toString() {
        return "Suite #" + this.getNumber();
    }
    
    public String bedType() {
        return HotelAccommodation.KING;
    }
}

1. Is GuestRoom an abstract or a concrete class?

2. In each of the following code segments, identify whether or not the compiler is going to report a syntax error. In case of an error briefly state the reason.
   a. HotelAccomodation ha = new HotelAccomodation();
   
   b. HotelAccomodation ha;
      ha = new GuestRoom("243", 165, HotelAccomodation.KING);
   
   c. Suite s = new Suite();
d. GuestRoom r = new HotelAccomodation();

e. HotelAccomodation ha;
   ha = new Suite("175", 300.0, 2);
   System.out.println("Suite " + ha.getNumber() + 
   " has " + ha.getNRooms() + "rooms");

f. HotelAccomodation ha = new GuestRoom("243", 165,
   HotelAccomodation. KING);

g. OnSale item = new GuestRoom();

h. OnSale item = new OnSale();

3. Will the following code run without an error?
   If yes, indicate what is printed when the following code segment is executed.

a. GuestRoom r = new GuestRoom();
   System.out.println(r.getPrice());
   System.out.println(r.bedType());
   System.out.println(r.askingPrice());

b. HotelAccomodation ha;
   ha = new Suite("175", 300, 2);
   System.out.println(ha instanceof HotelAccomodation);
   System.out.println(ha instanceof Object);
   System.out.println(ha instanceof Suite);
   System.out.println(ha instanceof GuestRoom);
   System.out.println(ha instanceof OnSale);
   System.out.println(ha.toString());

4. Show how to define an array that would be capable of storing up to 25 objects of
classes Suite and/or GuestRoom. Then, write a loop that would count the
number of GuestRooms and Suites in it.
Problem 3.
Create a definition for a concrete subclass ConferenceRoom of class HotelAccomodation from Problem 2.

The ConferenceRoom class should add instance variables capacity (int) to represent the maximum capacity of a conference room.

Define a 2-argument constructor of class ConferenceRoom that is passed the room number and maximum capacity, and assigns the fields of the ConferenceRoom object to the values of these parameters. The price field should be assigned according to the following rule: STANDARD_PRICE should be charged for each 2 persons. For example, if the capacity is 7, the price should be 3 * STANDARD_PRICE.

Implement all method(s) required by the abstract superclass of ConferenceRoom and no others. Note: A ConferenceRoom is a room with no beds.