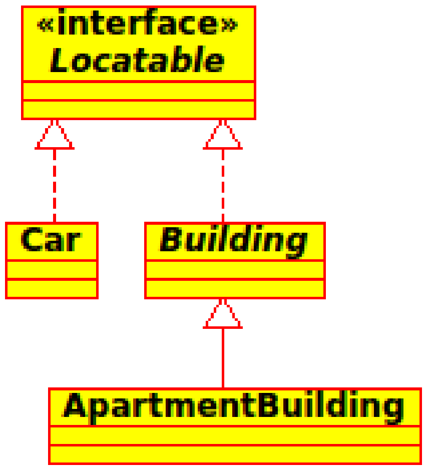
CIS 351 - Practice Midterm 2 Answers

Answers might be in different order

1. (6 points) Draw a UML diagram corresponding to the accompanying `Locatable` interface and the `Building`, `Car`, and `ApartmentBuilding` classes. For full credit, you must use correct UML "syntax", including arrow types, etc. You may use underlining in place of italics where appropriate. YOU DO NOT NEED TO INCLUDE INSTANCE VARIABLES OR METHODS IN YOUR DIAGRAM.



2. (15 points) Given the `Locatable` interface and the `Building`, `Car`, and `ApartmentBuilding` classes and the following declarations, which of the following will compile (C) or will not compile (N).

```
Object obj;
Locatable place;
Building bld;
Car auto;
```

(a) N `obj = new Locatable(38.443, -78.812);`
(b) C `auto = new Car("Chevy", 36.257, -81.321);`
(c) C `place = new ApartmentBuilding("91 E. Grace St.", 38.441, -78.871, 26);`
(d) C `bld = new ApartmentBuilding("91 E. Grace St.", 38.441, -78.871, 26);`
(e) N `bld = new Car("Chevy", 36.257, -81.321);`
(f) N `bld = new Building("701 Carrier Drive", 38.434,-78.863);`

Assuming that `place` contains an appropriate object, which of the following statements will compile(C) and which will not(N).

(g) N `place.longitude = 36.54;`
(h) C `double lat = place.getLatitude();`
(i) C `System.out.println(place.equals("some string"));`
(j) N `place.setLatitude(83.4);`

Assume that we are rewriting the `toString` method of the `ApartmentBuilding` class.

```
public String toString()
{
    String result = "";

    // Statements go here.
}
```

Which of the following statements would compile(C) and which will not(N) when included in this method?

(k) C `result += this.address;`
(l) C `result += getLongitude();`
(m) N `result += latitude;`
(n) C `result += getNumUnits();`
(o) N `result += Building.getAddress();`

3. (8 points) What output will be produced by the following program, `Driver.java`, given the code on the reference pages?

```
public class Driver
{
    public static void main(String[] args)
    {
        Building bld;
        ApartmentBuilding apt;
        bld = new ApartmentBuilding("18 Ohio", 37.441,
                                    -78.873, 26);
        apt = (ApartmentBuilding) bld;

        System.out.println(bld.toString());
        System.out.println(apt.toString());
        printInfo(bld);
        printInfo(apt);
    }

    public static void printInfo(Building place)
    {
        System.out.println("BUILDING: " + place.toString());
    }

    public static void printInfo(ApartmentBuilding place)
    {
        System.out.println("APARTMENT: " + place.toString());
    }

}
```

```
18 Ohio (26 units)
18 Ohio (26 units)
BUILDING: 18 Ohio (26 units)
APARTMENT: 18 Ohio (26 units)
```

4. Write a utility method named `arcticOnly` that takes an `ArrayList` of `Locateable` objects named `global`, and returns a new ArrayList of `Locateable` objects that only includes the the `Locateable`s from `global` that have a latitude greater than 66.56.

```
public static ArrayList<Locatable> arcticOnly(ArrayList<Locatable> global)
{
    ArrayList<Locatable> result;

    result = new ArrayList<Locatable>();

    for (Locatable loc : global)
    {
        if (loc.getLatitude() > 66.56)
        {
            result.add(loc);
        }
    }

    return result;
}
```

```
public interface Locatable
{
    public double getLatitude();
    public double getLongitude();
}
```

```
public class Car implements Locatable
{
    private String model;
    private double latitude;
    private double longitude;

    public Car(String model, double latitude, double longitude)
    {
        this.model = model;
        setLocation(latitude, longitude);
    }

    public String getModel()
    {
        return model;
    }

    public double getLatitude()
    {
        return latitude;
    }

    public double getLongitude()
    {
        return longitude;
    }

    public void setLocation(double latitude, double longitude)
    {
        this.latitude = latitude;
        this.longitude = longitude;
    }

}
```

```java
public abstract class Building implements Locatable
{
    protected String address;
    private final double latitude;
    private final double longitude;

    public Building(String address, double latitude, double longitude)
    {
        this.address = address;
        this.latitude = latitude;
        this.longitude = longitude;
    }

    public double getLatitude()
    {
        return latitude;
    }

    public double getLongitude()
    {
        return longitude;
    }

    public String getAddress()
    {
        return address;
    }

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

```java
public class ApartmentBuilding extends Building
{
    private int numUnits;

    public ApartmentBuilding(String address, double latitude,
                             double longitude, int numUnits)
    {
        super(address, latitude, longitude);
        this.numUnits = numUnits;
    }

    public int getNumUnits()
    {
        return numUnits;
    }

    public String toString()
    {
        return address + " (" + numUnits + " units)";

    }
}
```

5. Describe the worst case running time of the following code in “big-Oh”

a)

```
int f3(int n) {
     int sum = 73;
     for(int i=0; i < n; i++) {
          for(int j=i; j >= 5; j--) {
               sum--;
          }
     }
     return sum;
}
```

Big O = O(n2)

b)

```
void f2(int n) {
     for(int i=0; i < n; i++) {
          for(int j=0; j < 10; j++) {
               for(int k=0; k < n; k++) {
                    for(int m=0; m < 10; m++) {
                         System.out.println("!");
     }    }    }    }
}
```

Big O = O(n2)

6. (8 points) Given the attached `MotorVehicle`, `Truck`, and `Hybrid` classes, what will be printed when the following main method is executed?

```
    public static void main(String[] args)
    {
        MotorVehicle[] vehicles = new MotorVehicle[4];
        Truck truck;
        Hybrid prius;

        truck = new Truck(20, 25);
        vehicles[0] = truck;

        prius = new Hybrid(50, 10);
        prius.ride(100);
        vehicles[1] = prius;

        vehicles[2] = new Hybrid(40, 15);
        vehicles[2].ride(250);

        vehicles[1].ride(50);
        vehicles[0].ride(10);
        vehicles[3] = new Truck(15, 30);

        printVehicleStatus(prius);
        printVehicleStatus(truck);
        for (int i = 0; i < vehicles.length; i++)
        {
            printVehicleStatus(vehicles[i]);
        }

    }

    public static void printVehicleStatus(Truck pickup)
    {
        System.out.println("Truck Status: " + pickup.toString());
        System.out.println("Fuel Left: " + pickup.getCurrentFuel());
    }

    public static void printVehicleStatus(MotorVehicle motorVehicle)
    {
        System.out.println("Vehicle Status: " + motorVehicle.toString());
        System.out.println("Fuel Left: " + motorVehicle.getCurrentFuel());
    }
```

```
Salary Pay: Lovelace, Ada: 20000.0
Hourly Pay: Hopper, Grace: 45.0
Employee Pay: Lovelace, Ada, 20000.0
Employee Pay: Hopper, Grace, 45.0
```

7. Assume that LL is a DOUBLY linked list with the head node and at least one other internal node M which is not the last node. Write few lines of code to accomplish the following. You may assume that each node has a next pointer and prev pointer. You may NOT swap data to accomplish any of the following operations. For each operation, assume the original list as described above. You are encouraged to draw pictures to justify your code. Note that for each operation, you need to manipulate at least two pointers, next and prev.

a. Delete the head node

```
head = head.next;
head.prev = null;
```

b. Insert a node P immediately after M

```
P.next = M.next;
M.next = P;
(P.next).prev = P;
P.prev = M;
```