Examine the following three classes, named Employee, SalaryEmployee, and Manager:

```
public abstract class Employee{

   private String name;
   private int years;
   private int BASE_DAYS;

   public Employee(String name, int years){
      this.name = name;
      this.years = years;
      BASE_DAYS = 10;
   }

   public int getYears(){
      return years;
   }

   public double vacationDays(){
      double days = BASE_DAYS;
      if(years > 20){
        days += 15;
      }
      else if(years > 10){
        days += 10;
      }
      return days;
   }

   public abstract double pay();
}
```

```
public class SalaryEmployee extends Employee{

      private double baseSalary;

      public SalaryEmployee(String name, int
                        years, double base){
         super(name, years);
         baseSalary = base;
      }

      public double pay(){
         return baseSalary * getYears()/2.5;
      }
}
```

```
public class Manager extends SalaryEmployee{

  private String department;

  public Manager(String name, int years,
            double base, String dept){
     super(name, years, base);
     department = dept;
  }

  public double vacationDays(){
      return super.vacationDays() * 1.5;
  }

  public double expenseFunds(){
     double funds = 500;
     if(getYears() > 5){
         funds *= 1.5;
     }
     return funds;
  }
}
```

If lines of code shown in a) – e) below above were included in the main method of a driver program (saved in the same folder as `Employee`, `SalaryEmployee`, & `Manager`), would they be valid or invalid? If you answered valid, you must also write the output (what would be printed), OR if you answered invalid, you must clearly state the reason.

a)
```
SalaryEmployee emp = new SalaryEmployee("Susan Apple", 12, 50000);
System.out.println(emp.pay());
```

b)
```
Employee emp = new Employee("Fred Peach", 9);
System.out.println(emp.getYears());
```

c)
```
SalaryEmployee emp = new SalaryEmployee("Simon Pear", 6, 50000);
System.out.println(emp.expenseFunds());
```

d)
```
Manager emp = new Manager("Lisa Grape", 15, 70000, "Sales");
System.out.println(emp.vacationDays());
```

e)
```
Manager emp = new Manager("Tim Banana", 5, 60000, "Sales");
System.out.println(emp.pay());
```

2. (8 marks) Please complete the following static method so that it meets the requirements discussed below.

The method will take a character (***letter***) and an array of Strings (***words***) as a parameter. Assume ***letter*** could be an uppercase or lowercase letter and ***words*** will not be null. The method must return an integer array of 3 elements where:

- the 0th element is the number of Strings in ***words*** that begin with a character that comes before ***letter*** in the alphabet
- the 1st element is the number of Strings in ***words*** that begin with the same character as ***letter***
- the 2nd element is the number of Strings in ***words*** that begin with a character that comes after ***letter*** in the alphabet

Strings in ***words*** may start with a lowercase letter, uppercase letter, or non-alphabetic character. If a String in ***words*** starts with a non-alphabetic character, do not increment any of the three counters.

```
public static int[] alphaPlaceCount(char letter, String[] words){
```

```java
public interface Activity {
      public void play();
}
}
```

```java
public class Race implements Activity {
      public Race() { }
      public void play() {
            System.out.println("run!");
      }
      public void duration(){
            System.out.println("5
minutes");
      }
}
```

```java
public class FacePaint implements
Activity{
      public FacePaint() { }
      public void play() {
            System.out.println("paint
face");
      }
      public void ageRange() {
            System.out.println("3-8
years”);
      }
}
```

```java
public class SackRace extends Race {
      public SackRace() { }
      public void play() {
            System.out.println("hop!");
      }
      public void duration(){
            System.out.println("1
minute");
      }

}
```

If the following lines of code appeared in the main method of a driver program, would the program compile and run? (yes or no?) If no, write 1-2 sentences to explain why not. If yes, indicate what the output would be.

a)

```java
Activity obj1 = new FacePaint();
  obj1.ageRange();
```

b)

```java
Activity obj3;
Race obj4 = new SackRace();
obj3 = obj4;
((SackRace)obj3).duration();
```

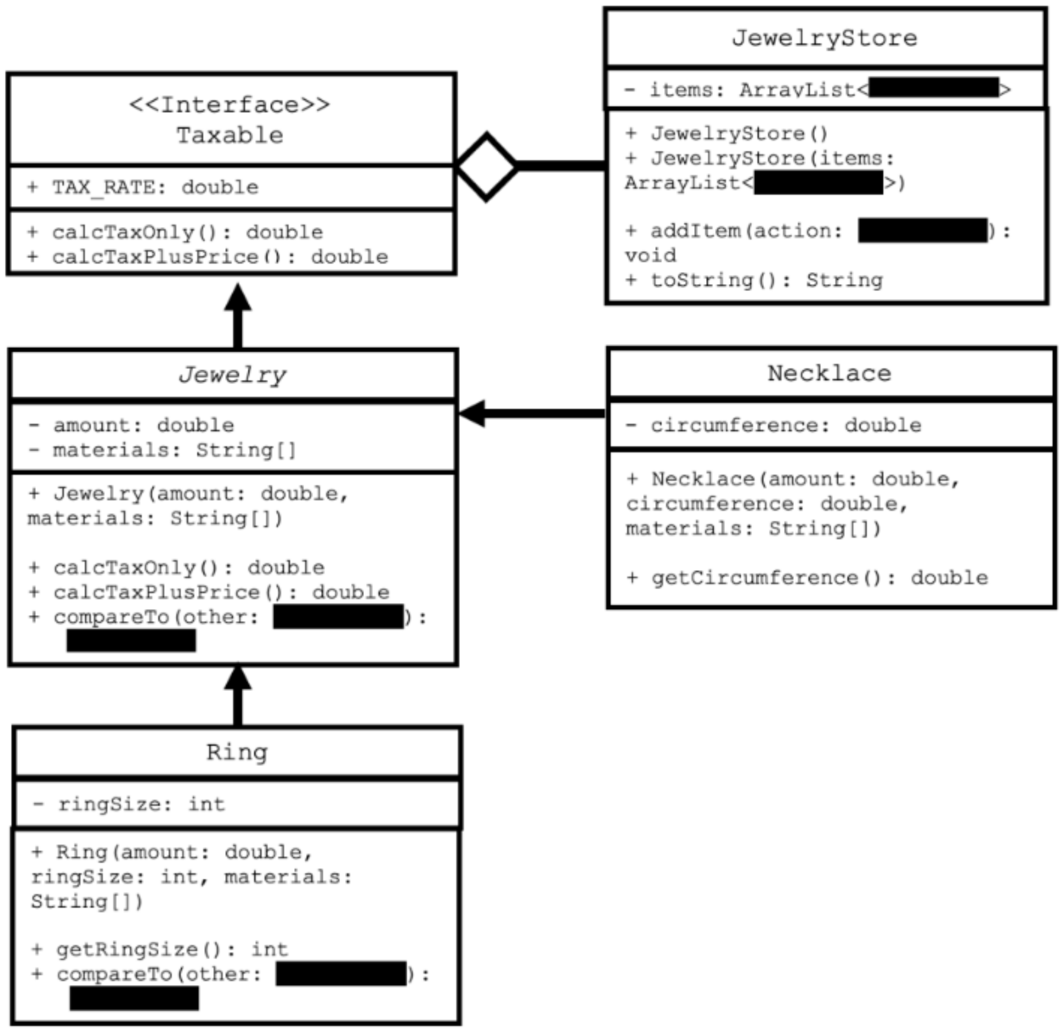
c)

```java
Race obj1 = new SackRace();
obj1.duration();
```

PART B – Programming

4. **[Value: 17]** The following UML Diagram represents four Java classes and one Java interface. Some data types of the Diagram are blacked out. Note that the spaces blacked out are not to scale meaning that the actual keywords may be longer or shorter than the provided blacked out space.

   In the following questions, you will be provided with code representing the UML Diagram. The code will have various blanks that you must fill in so the code will compile and run appropriately matching the requirements laid out in the UML Diagram. All methods should follow appropriate Object Oriented Programming guidelines discussed in class.


JewelryStore
- items: ArrayList<[redacted]>
+ JewelryStore()
+ JewelryStore(items: ArrayList<[redacted]>)
+ addItem(action: [redacted]): void
+ toString(): String
<<Interface>> Taxable
+ TAX_RATE: double
+ calcTaxOnly(): double
+ calcTaxPlusPrice(): double
Jewelry
- amount: double
- materials: String[]
+ Jewelry(amount: double, materials: String[])
+ calcTaxOnly(): double
+ calcTaxPlusPrice(): double
+ compareTo(other: [redacted]): [redacted]
Necklace
- circumference: double
+ Necklace(amount: double, circumference: double, materials: String[])
+ getCircumference(): double
Ring
- ringSize: int
+ Ring(amount: double, ringSize: int, materials: String[])
+ getRingSize(): int
+ compareTo(other: [redacted]): [redacted]

a) **[Value: 3]** The **Taxable** Interface. Fill in 3 blanks.

```
public interface Taxable extends___________ Comparable<_Jewelry_________>{

      public _final_________ double TAX_RATE = 0.15;

      public double calcTaxOnly();
      public double calcTaxPlusPrice();
}
```

b) **[Value: 5] Jewelry.java**. Fill in 4 blanks. Jewelry should be compared based on the price + tax.

```
public abstract class Jewelry implements ____________________{
      private double amount;
      private String[] materials;

      public Jewelry(double amount, String ... materials){
            this.amount = amount;
            this.materials = materials;
      }

      public double calcTaxOnly(){
            return amount * Jewelry.TAX_RATE;
      }

      public double calcTaxPlusPrice(){
            return amount + this.calcTaxOnly();
      }

      public int_______ compareTo(Jewelry___________ other){
            if (calcTaxPlusPrice() == other.calcTaxPlusPrice()) {
            return 0;
            } else if (calcTaxPrice() > other.calcTaxPlusPrice()) {
            return 1;
            ] else {
            return -1;
            ]
            ____________________________________________
            ____________________________________________
            ____________________________________________
            ____________________________________________
            ____________________________________________
            ____________________________________________
      }
}
```

c) **[Value: 0.5] Necklace.java**. Fill in 1 blank.

```
public class Necklace __extends Jewelry____________________{
      private double circumference;

      public Necklace(double amount, double circumference, String ...
                  materials){
            super(amount, materials);
            this.circumference = circumference;
      }

      public double getCircumference(){
            return circumference;
      }
}
```

d) **[Value: 5] Ring.java**. Fill in 4 blanks. Rings should be ordered based on their price + tax and then their ring size, i.e., if two rings have the same price, then compare the ring sizes.

```
public class Ring extends Jewelry__________________{
      private int ringSize;

      public Ring(double amount, int ringSize, String ... materials){
            super(amount, materials);
            this.ringSize = ringSize;
      }

      public int getRingSize(){
            return ringSize;
      }

      public int_______________ compareTo(Taxable__________ other){

            ____________________________________________________________

            ____________________________________________________________

            ____________________________________________________________

            ____________________________________________________________

            ____________________________________________________________

            ____________________________________________________________

            ____________________________________________________________

            ____________________________________________________________

            ____________________________________________________________

            ____________________________________________________________

            ____________________________________________________________

            ____________________________________________________________

      }
}
```

e) **[Value: 3.5] JewelryStore.java**. Fill in 7 blanks.

```
import java.util.ArrayList;

public class JewelryStore{
      private ArrayList<________________________> items;

      public JewelryStore(){
            items = null;
      }

      public JewelryStore(ArrayList<________________________> items){
            this.items = items;
      }
```

```
	//Method should add a passed in item to the ArrayList

	public void addItem(_________________________ item){
		if(items == null){

			items = __________________________________________;

		}
		items.add(item);
	}

	//Method should print the values of items in the ArrayList
	public String toString(){
		if(items == null){
			return "No items in the store";
		}
		else{
			String toReturn = "";

			for(____________________________ : ______________){

				toReturn += _______________.calcTaxPlusPrice()
				            + "\n";
			}
			return toReturn;
		}
	}
}
```