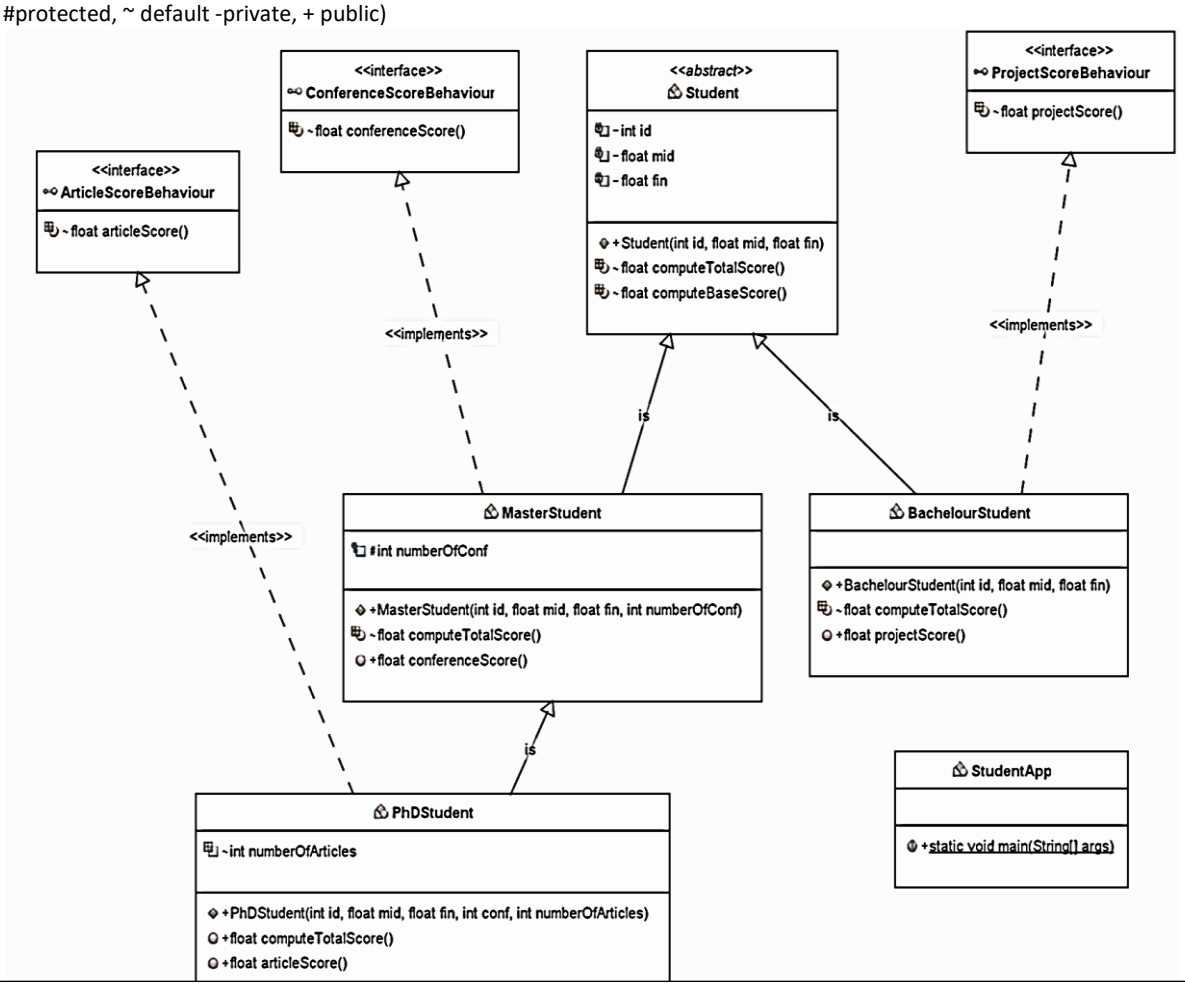
FULL NAME:	
ID:	DAY ( ) NIGHT ( )

SCORE	Q1	Q2	Q3	Q4	TOTAL

OBJECT ORIENTED PROGRAMMING | FINAL EXAM 11.01.2022 | 11.00 | 100 mins

Q1 (35p): Write the classes and interfaces given by following diagram. Override methods for each class given by table
(symbols: #protected, ~ default -private, + public)



	computeTotalScore()	computeBaseScore()	projectScore()	conferenceScore()	articleScore()
Student	*abstract*	Mid * 0.4 + fin * 0.6	--	--	--
BachelourStudent	computeBaseScore() + projectScore()	--	20	--	--
MasterStudent	computeBaseScore() + conferenceScore()	--	--	numberOfConf * 5	--
PhDStudent	computeTotalScore() + articleScore()	--	--	--	numberOfArticle*8

Q2(25p): Using the relationship given UML in the previous question and statements given in main method, check if the following compiles, runs, and make it work if possible.

Q2.Önceki sorudaki UML ile verilen ilişkiyi ve yanda mainde verilen ifadeleri kullanarak, aşağıdaki tabloda verilen ifadelerin derlenip çalıştığını kontrol ediniz ve mümkünse düzeltin. Tam puan almak icin cevabinizi aciklayin.

```
public class TestMain {
    public static void main(String[] args){
       Student student1 = new BachelourStudent(10, 50, 90);
       Student student2 = new MasterStudent(10, 50, 90,5);
       Student student3 = new PhDStudent(10, 50, 90,2,1);
   }}
```

Statement	Compile?	Run?	If compiles or runs, explain why? If not, correct it (if it is possible)
`student1.projectScore();`	NO	NO	you need to downcast ((BachelourStudent)student1).projectScore();
`((ConferenceScoreBehaviour)student3).conferenceScore();`	YES	YES	
`((ArticleScoreBehaviour)student2).articleScore();`	YES	NO	MasterStudent cannot be cast to ArticleScoreBehaviour (use student3)
`student2.articleScore();`	NO	NO	articleScore method not exist in MasterStudent (use (PhDStudent)student3)
`((ProjectScoreBehaviour)student3).projectScore();`	YES	NO	PhDStudent cannot be cast to ProjectScoreBehaviour (use student1)

Q3(30p). Write a class called BankAccount. The class must have 3 attributes: accountNo(**int**), holderName(**String**) and balance(**double**).

a) Class constructor will have to set these 3 attributes.

b) Create a balanceChange(double amount) method to reduce balance value by given amount.

c) In the Main method:

1. Create 4 objects (a1, a2, a3, a4) from this class and add them all to an ArrayList called myAccounts. List will only accept BankAccount type.
2. Sort objects by holderName in the list. **(Hint: implement Comparable interface)**
3. Reduce the balances by 500 for all the accounts and print sorted objects in the list.

BankAccount adında bir class yazın. Sınıfın 3 özelliği olmalıdır: accountNo(int), holderName(String) ve balance(double).

a) Constructor un bu 3 özelliği ayarlaması (set etmesi) gerekecektir.

b) balabce değerini verilen miktar kadar azaltmak için bir balanceChange(double amount) metodu oluşturun.

c) main metodda:

1. Bu classtan 4 nesne (a1, a2, a3, a4) oluşturun ve hepsini myAccounts adlı bir ArrayList'e ekleyin. Liste yalnızca BankAccount türünü kabul edecektir.

2. Listedeki nesneleri holderName'e göre sıralayın. (İpucu: Comparable interface'ni uygulayın)

3. Tüm hesaplar için balance değeri 500 azaltın ve listedeki sıralı nesneleri yazdırın.

Q4(20p). Write outputs in the given table. All the classes are declared in the same package.

```
public class Computer {
    static int counter;
    int year;
    String model;
    public Computer(int year, String model) {
        this.year = year;
        this.model = model;
        counter++;
    }
    @Override
    public String toString() {
        return model+year;
    }
}

class Desktop extends Computer{
    int id;
    int memorySize;
    public Desktop(int year, String model) {
        super(year, model);
        this.id=counter;
    }
    public void setMemory(int memorySize){
        this.memorySize=memorySize;
    }
    @Override
    public String toString() {
        return id+". "+super.toString()+". Memory:"+memorySize;
    }
}

class Laptop extends Computer{
    int memorySize;
    int id;
    public Laptop(int year, String model) {
        super(year, model);
        this.id=counter;
    }
    public void setMemory(int memorySize){
        this.memorySize=memorySize;
    }
     @Override
    public String toString() {
        return id+". "+super.toString()+". Memory:"+memorySize;
    }
}

public class ExamMainClass {

    public static void main(String[] args) {
        Computer c1= new Desktop(2019,"Dell XPS");
        ((Desktop)c1).setMemory(16);
        Computer c2= new Laptop(2020,"MacBook");
        ((Laptop)c2).setMemory(32);
        Computer c3= new Desktop(2017,"iMac");
        ((Desktop)c3).setMemory(16);
        Computer[] computers={c1,c2,c3};

        for(Computer c:computers)
          System.out.println(c); // (o1 to o3)

        System.out.println(Computer.counter); // (o4)
        System.out.println(((Laptop)computers[1]).id); // (o5)
    }
}
```

(o1)	1. Dell XPS2019. Memory :16	(o4)	3
(o2)	2. MacBook2020. Memory :32	(o5)	2
(o3)	3. iMac2017. Memory :16		

Q1: Answer:

```
package studentapp;

public class StudentApp {
   public static void main(String[] args)
   {
     Student student1 = new BachelourStudent(10, 50, 90);
     Student student2 = new MasterStudent(10, 50, 90,5);
     Student student3 = new PhDStudent(10, 50, 90,2,1);
   }
}

//////////////////////////////////////
public abstract class Student {
private int id;
private float mid;
private float fin;

   public Student(int id, float mid, float fin) {
      this.id = id;
      this.mid = mid;
      this.fin = fin;
   }

   abstract float computeTotalScore();

   float computeBaseScore(){
      return mid*0.4f + fin * 0.6f ;
   }
}

//////////////////////////////////////

interface ProjectScoreBehaviour {
    float projectScore();
}

interface ConferenceScoreBehaviour{
   float conferenceScore();
}

interface ArticleScoreBehaviour{
   float articleScore();
}

//////////////////////////////////////

public class BachelourStudent extends Student implements ProjectScoreBehaviour{

   public BachelourStudent(int id , float mid, float fin) {
      super(id, mid, fin);
   }
```

```
    @Override
    float computeTotalScore() {
     return super.computeBaseScore()+ this.projectScore();
    }

    @Override
    public float projectScore() {
        return 50;
    }

}

//////////////////////////////////////////
public class MasterStudent extends Student implements ConferenceScoreBehaviour{
    protected int numberOFConf;
    public MasterStudent(int id, float mid, float fin, int numberOfConf) {
        super(id, mid, fin);
        this.numberOFConf = numberOfConf;
    }

    @Override
    float computeTotalScore() {
        return super.computeBaseScore() + this.conferenceScore();
    }

    @Override
    public float conferenceScore() {
        return numberOFConf*5;
    }

}

//////////////////////////////////////////////////////

public class PhDStudent extends MasterStudent implements ArticleScoreBehaviour {

    int numberOfArticles;
    public PhDStudent(int id, float mid, float fin, int conf, int numberOfArticles) {
        super(id, mid, fin, conf);
        this.numberOFConf = numberOfArticles;
    }

    @Override
    public float computeTotalScore(){
        return  super.computeTotalScore() + this.articleScore();
    }

    @Override
    public float articleScore() {
        return numberOfArticles*8;
    }

}
```

Q3:answer

```
package bank;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Iterator;

public class Bank {

   public static void main(String[] args) {
      ArrayList<BankAccount> accounts=new ArrayList<>();
      accounts.add(new BankAccount(101,"Basem",30000));
      accounts.add(new BankAccount(102,"Ali",50000));
      accounts.add(new BankAccount(103,"Omer",40000));
      accounts.add(new BankAccount(104,"Ahmed",20000));
      Collections.sort(accounts);
      for(BankAccount a: accounts){
         a.balanceChange(5000);
         System.out.println(a);
      }

   }

}

////////////////////////////////////
public class BankAccount implements Comparable<BankAccount>{
int accountNo;
String holderName;
double balance;
   public BankAccount(int accountNo, String holderName, double balance) {
      this.accountNo = accountNo;
      this.holderName = holderName;
      this.balance = balance;
   }
void balanceChange(double amount){
   this.balance-=amount;
}

   @Override
   public int compareTo(BankAccount t) {
      return this.holderName.compareTo(t.holderName);

      }

   @Override
   public String toString() {
      return "Account:"+accountNo+", "+holderName+", "+balance;
   }
}
```