King Saud University

College of Computer and Information Sciences

Computer Science Department

Course Code:	CSC 113
Course Title:	Computer Programming II
Semester:	Spring 2019
Exercises Cover Sheet:	**Final Exam**

Student Name:	
Student ID:	
Student Section No.	

Tick the Relevant	Computer Science B.Sc. Program ABET Student Outcomes	Question No. Relevant Is Hyperlinked	Covering %
X	**a) Apply knowledge of computing and mathematics appropriate to the computer science;**		
	b) Analyze a problem, and identify and define the computing requirements appropriate to its solution		
X	**c) Design, implement and evaluate a computer-based system, process, component, or program to meet desired needs;**		
X	**d) Function effectively on teams to accomplish a common goal;**		
	e) Understanding of professional, ethical, legal, security, and social issues and responsibilities;		
	f) Communicate effectively with a range of audiences;		
	g) Analyze the local and global impact of computing on individuals, organizations and society;		
	h) Recognition of the need for, and an ability to engage in, continuing professional development;		
X	**i) Use current techniques, skills, and tools necessary for computing practices.**		
	j) Apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices;		
	k) Apply design and development principles in the construction of software systems of varying complexity;		

Exercise 1:

Give the output of the following program using the following input data: 12, 5, 5.

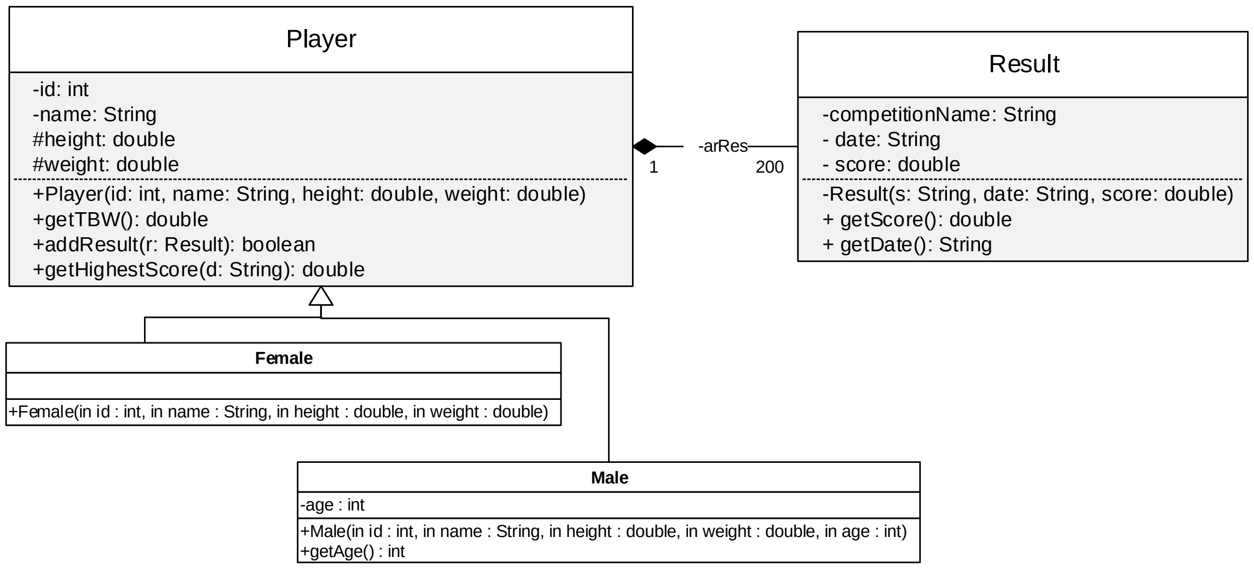
```
public class Vehicle {
            protected String brand;
            protected double price;
            protected int nbOfSeats;

            public Vehicle() { brand = "Unknown";          price       = 50.0;            nbOfSeats = 10;
                        System.out.println(" .... Brand : " + brand + " --- Price : " + price + " --- Seats : " + nbOfSeats);
            }
            public Vehicle(String b, double p, int n) { brand = b;                    price      = p;          nbOfSeats = n;
                        System.out.println(" .... Brand : " + brand + " --- Price : " + price + " --- Seats : " + nbOfSeats);
            }
            public void show() {  System.out.println(" .... Brand : " + brand + " --- Price : " + price + " --- Seats : " + nbOfSeats);       }
}
public class Bus extends Vehicle {
            private String name;
           private int nbOfPassengers;

            public Bus(){ name = "Hafeela";        nbOfPassengers = 0;
                        show();
            }
            public Bus(String s, String b,  double p, int n) {  name = s;              brand = b;                price = p;
                        nbOfSeats = n;        nbOfPassengers = 0;
                        show();
            }
            public void show() {
                        super.show();
                        System.out.println(" **** Name : " + name + " .... Nb of Passengers : " + nbOfPassengers);
            }
            public void addPassangers(int nb) throws Exception{
                        if (nb <= 0 )   throw new Exception (“Invalid Number Of Passengers”);
                        if (nb > (nbOfSeats – nbOfPassengers) )   throw new Exception (“Parameter value exceeds available seats”);
                        nbOfPassengers  += nb ;
                        show();
            }
            public void dropPassangers(int nb) {
                       if ( nb <= 0 || nb > nbOfPassengers )   throw new Exception (“Invalid Number Of Passengers to drop”);
                       nbOfPassengers  -= nb ;
                       show();
             }
}
```

```
public class Program {
            public static void main(String[] args) { Scanner in = new Scanner(System.in);
                        Bus m1 = new Bus();
                        Bus m2 = new Bus("B2", "Mercedes", 70.0, 5);
                         for (int i = 0; i < 3; i++) {
                                try {
                                     if (i%2 == 0) m1.addPassangers(in.nextInt());
                                     else m2.dropPassangers(in.nextInt());
                              }
                              catch(Exception e) { System.out.println (e.getMessage());
                        }
            }
}
```

Exercise 2:


Player
-id: int
-name: String
#height: double
#weight: double
+Player(id: int, name: String, height: double, weight: double)
+getTBW(): double
+addResult(r: Result): boolean
+getHighestScore(d: String): double
-arRes
1
200
Result
-competitionName: String
- date: String
- score: double
-Result(s: String, date: String, score: double)
+ getScore(): double
+ getDate(): String
Female
+Female(in id : int, in name : String, in height : double, in weight : double)
Male
-age : int
+Male(in id : int, in name : String, in height : double, in weight : double, in age : int)
+getAge() : int

The class Player:

- Attributes:
  - ***id:*** identifier of the Player.
  - ***name***: name of the Player.
  - ***height***: height (in cm) of the Player.
  - ***weight:*** weight (in kilogram) of the Player.
- Methods:
  - ***Player (…)***: constructor.
  - ***getTBW( ):*** calculates the Total Body Water (TBW) of the Player based on the following formulas:
    - For Male: TBW = 2.447 - (0.09156 * age) + (0.1074 * height) + (0.3362 * weight)
    - For Female: TBW = -2.097 + (0.1069 * height) + (0.2466 * weight)
  - ***addResult(r:** Result)***: It** adds the result ***r*** to the player results. If the Result's score is negative, it raises an ***InvalidParameterException*** with message "*Invalid Score*". If the array of results is full, this method throws an ***ArrayOutOfBoundsException*** with message "*Array is Full*". It returns true if the insertion is done successfully. Otherwise, it returns false.
  - ***getHighestScore(d: String*)**:** this method returns the score of the best Result that the player achieved on day ***d***.

The class Male:

- Attributes:
  - ***age:*** age of the Male Player.
- Methods:
  - ***Male (…)***: constructor.
  - ***getAge( ):*** It returns the age of the Male Player

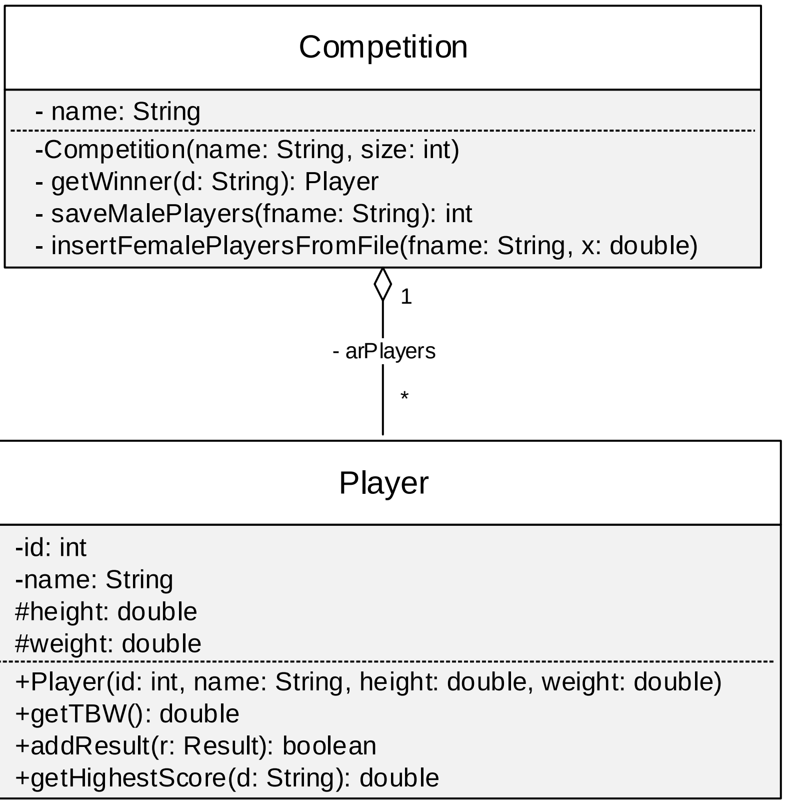
The class Result:
- Attributes:
  - ***date:*** date of achieving the Result.
  - ***score:*** score of the result.
- Methods:
  - ***Result (…)***: constructor.
  - ***getScore( ):*** It returns the score of the Result.
  - ***getDate( ):*** It returns the date of the Result.

QUESTION: Translate into Java code:

1. The class ***Result***,
2. The class ***Player***
3. The class ***Male.***

Exercise 3:

Let's consider the same class ***Player*** and its subclasses as described in Exercise 2.


Competition
- name: String
-Competition(name: String, size: int)
- getWinner(d: String): Player
- saveMalePlayers(fname: String): int
- insertFemalePlayersFromFile(fname: String, x: double)
1
- arPlayers
*
Player
-id: int
-name: String
#height: double
#weight: double
+Player(id: int, name: String, height: double, weight: double)
+getTBW(): double
+addResult(r: Result): boolean
+getHighestScore(d: String): double

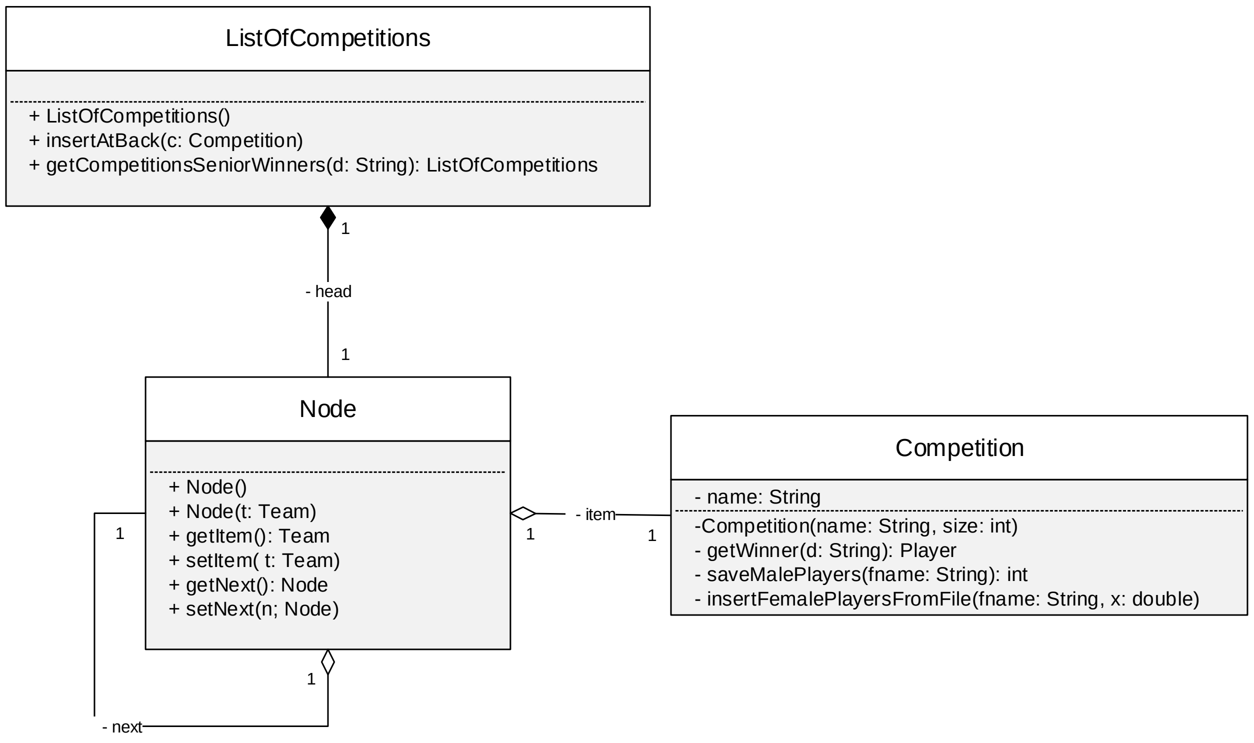
The class Competition:
- Attributes:
  - ***name:*** name of the Competition.
- Methods:
  - ***Competition (…)***: constructor.

- ***getWinner(d: String)*:** It returns the Player who achieved the highest score on day ***d***.
- ***saveMalePlayers(fname:*** String ***):*** It stores all Male Players in the file ***fname***. It returns the number of Male Players saved into the file.
- ***insertFemalePlayers(fname:*** String, ***x***: double): It receives a File of Players ***fname***. It adds to the Competition all Female Players having a TBW greater or equal than ***x***.

QUESTION: Translate into Java code the class ***Competition.***

Exercise 4:

Let's consider the class ***Competition*** as described in exercise 3.


ListOfCompetitions
+ ListOfCompetitions()
+ insertAtBack(c: Competition)
+ getCompetitionsSeniorWinners(d: String): ListOfCompetitions
1
- head
1
Node
+ Node()
+ Node(t: Team)
+ getItem(): Team
+ setItem(t: Team)
+ getNext(): Node
+ setNext(n; Node)
1
- item
1
Competition
- name: String
-Competition(name: String, size: int)
- getWinner(d: String): Player
- saveMalePlayers(fname: String): int
- insertFemalePlayersFromFile(fname: String, x: double)
1
1
- next

The class ListOfCompetitions:

- Methods:
  - ***ListOfCompetitions ( )***: constructor.
  - ***insertAtFront(c: Competition*)**:** this method inserts the Competition ***c*** in the front of the List.
  - ***getCompetitionsSeniorWinners(d:*** String): this method returns a List that contains all Competitions where the winners on day ***d*** are Male and 30 years old.

```
import java.io.*;

public class Competiton {
private String name;
player arrplayers[];
private int counter;
public Competiton(String nsme, int size) {
     arrplayers= new player[size];
     name=nsme;
     counter=0;
}

public player getwinner(String d) {
     int max=0;
               for(int i=0;i<counter;i++)

     if(arrplayers[i].gethaihestScore(d)>arrplayers[max].gethaihestSco
re(d))
                         max=i;
               return arrplayers[max];
}

public int savemaleplayer(String filename) throws IOException{
     File f= new File(filename);
     FileOutputStream outcheck= new FileOutputStream(f);
     ObjectOutputStream theoutcheck = new
ObjectOutputStream(outcheck);
     int n=0;
     for(int i=0;i<counter;i++)
     if(arrplayers[i]instanceof Male) {
          theoutcheck.writeObject(arrplayers[i]);
     n++;}
     theoutcheck.close();
     outcheck.close();
     return n;
}
public void insertmaleplayers(String filename, double x) throws
IOException {

     File f= new File(filename);
     FileInputStream instream=new FileInputStream(f);
     ObjectInputStream theobj= new ObjectInputStream(instream);
     try{
          while(true){
          try{
               player p= (player) theobj.readObject();
               if(p instanceof Male)
                    arrplayers[counter++]= (player) p;
          }
          catch(ClassNotFoundException e){
               System.out.println(e);
               }}
```

```
	}
	catch(EOFException e){
		System.out.println("Finished reading");
		}
		theobj.close();
		instream.close();

}




}









public class ListOfCompetitons {
	Node head;

	ListOfCompetitons(){
		head=null;
	}
	public void insertatback(Competiton c) {
		Node newone= new Node(c);
		Node current= head;
		if(head==null)
		head=newone;
		else
			while(current.getNext()!=null) {
				current=current.getNext();
		current.setNext(newone);}

	}

	ListOfCompetitons getit(String d) {
		Node current= head;
		ListOfCompetitons s= new ListOfCompetitons();
		while(current!=null)
			if(current.getData().getwinner(d)instanceof Male)

	if(((Male)current.getData().getwinner(d)).getAge()>30)
```

```
                                s.insertatback(current.getData());

          return s;



     }

}
```

```
public class Male extends player  {
     private int age;

     public Male(int id, String name, double weghitSe, double hight,
int age) {
          super(id, name, weghitSe, hight);
          this.age = age;
     }

     public int getAge() {
          return age;
     }

     @Override
     public double getTBW() {

          return 2.447-
(0.09156*age)+(0.1074*hight)+(0.3362*weghitSe);
     }

}
```

```
public class Node {
	private Competiton data;
	private Node next;

	Node(Competiton C){
		data=C;
		this.next= null;
	}

	public Competiton getData() {
		return data;
	}

	public void setData(Competiton data) {
		this.data = data;
	}

	public Node getNext() {
		return next;
	}

	public void setNext(Node next) {
		this.next = next;
	}


}
```

```
import java.io.Serializable;
import java.security.InvalidParameterException;

public abstract class player implements Serializable {
	private int id;
	private String name;
	protected double weghitSe;
	protected double hight;
	Result arRes[];
	private int counter;
	public player(int id, String name, double weghitSe, double hight)
{
		this.id = id;
		this.name = name;
		this.weghitSe = weghitSe;
		this.hight = hight;
		arRes= new Result[200];
		counter=0;

	}

	public abstract double getTBW();


	public boolean addresult( Result r) throws Exception {
		if(arRes.length>=counter) {
			throw new ArrayIndexOutOfBoundsException("full");
		}
		if(r.getScore()<0) {
			throw new InvalidParameterException("invaild score");
		}
		if(arRes.length<counter&&r.getScore()>=0) {
			arRes[counter++]= new Result(r);
		return true;}
		return false;



	}
	public double gethaihestScore (String d) {
		double max=0;
		for(int i=0; i<counter;i++)

	if(arRes[i].getScore()>max&&arRes[i].getDate().equalsIgnoreCase(d
))
				max= arRes[i].getScore();
		return max;


	}



}
```

```
public class Result {
     private String competionName;
     private String date;
     private double score;
     public Result(String competionName, String date, double score) {
          super();
          this.competionName = competionName;
          this.date = date;
          this.score = score;
     }
     public Result(Result r) {
          this.competionName = r.competionName;
          this.date = r.date;
          this.score = r.score;

     }
     public String getDate() {
          return date;
     }
     public double getScore() {
          return score;
     }



}
```