Letterkenny Institute of Technology

Course code: OOPR CP603

YEAR 2 COMPUTING

(Common paper for all streams)

Subject:	Object Oriented Programming	**Stage:**	2
Date:	January 2019	**Examiners:**	Mr. T. Devine Dr. A. Belatreche
Time allowed:	3 hours		

INSTRUCTIONS

Answer any FOUR questions.

NOTE: It may be useful to remove the appendices from this paper for easy reference.

Question 1

Appendix A has partial code that implements a basic grid-based Noughts and Crosses game that was introduced in the module.

(a) In the `Board` class declare and create a 2 dimensional (3x3) class array called `board` to store 9 integers and provide the code in `clearBoard()` that initialises each value in the array to 0 (ZERO).
(8 marks)

(b) Identify any shared class variable(s) and method(s) that both `Human` and `Computer` classes use.
(3 marks)

(c) Write the code for a superclass called `Player` to implement the shared variable(s) and method(s) identified in (b).
(7 marks)

(d) Rewrite the signatures for the `Human` and `Computer` classes to become subclasses of the `Player` class.
(2 marks)

(e) Write an `AdvancedComputer` class which is a subclass of `Computer` and override the `setMove()` method using the overrides annotation. There is no need to provide code inside the overridden method.
(5 marks)

Question 2

Appendix B has a partial implementation of the `Manager` class.

(a) Provide the code for a `toString()` method to print variable data.
(6 marks)

(b) Modify the class so the tester code below will work:

```
Manager m1 = new Manager("Pep",5000000);

if(m1.equals(new Manager("Pep",5000000)))
  println("equals");
else
  println("!equals");
```

(11 marks)

(c) Modify the class so the tester code below will work:

```
Manager m1 = new Manager("Pep",5000000);
Manager m2 = new Manager("Jose",2500000);

if(m1.compareTo(m2)==0)
  println("m1 salary = m2 salary");
else if(m1.compareTo(m2)>0)
  println("m1 salary > m2 salary");
else
  println("m1 salary < m2 salary");
```

(8 marks)

Question 3

Appendix C shows a class diagram for the `Team`, `Player` and `Manager` classes. Provide all the Java code to implement the `Team` class. Examine the tester code in Appendix C(b) to help implement the correct solution.

(25 marks)

Question 4

Examine the `Player` and `Manager` classes in Appendix C.

(a) Provide all the code for a new parent class `Person` for `Player` and `Manager` that implements *shadow* class variables and methods.

(12 marks)

(b) Rewrite all the code in both `Player` and `Manager` classes to appropriately use the new parent class.

(10 marks)

(c) Describe the purpose of the `super` keyword.

(3 marks)

Question 5

```
int[] list = {26, 17, 5, 2};
```

(a) Given the array `list` above, how many *passes* are required for a selection sort algorithm to sort the values in ascending order.

(3 marks)

(b) Using `list` clearly show the state of the array after each *pass* of a selection sort algorithm used to sort the array values in ascending order.

(6 marks)

(c) Appendix D contains code for a selection sort. Provide the missing code in the method `indexOfLargestElement().`

(6 marks)

(d) For the `sequentialSearch()` method in Appendix D, provide the missing code that implements a sequential search algorithm.

(10 marks)

Question 6

Describe using code examples each of the following concepts:

(a) `try catch` statement
(b) `instanceof` operator
(c) `throw` keyword
(d) `extends` keyword
(e) default access modifier

(25 marks)

Appendix A

```
//
// Board.java
//
public class Board
{
  // Q1(a)
  // declare a 2d board array
  . . .

  public Board()
  {
    // Q1(a)
    // create 2d board array & clear the board
    . . .
    clearBoard();
  }


  // clear (0) all values in board array
  public void clearBoard()
  {
    // Q1(a)
    . . .
    . . .
    . . .
  }

}
```

Appendix A continued

```
//
// Human.java
//
public class Human // Q1(d)
{
  private int[] move = new int[2];

  public Human()
  {
    System.out.println("Human Player created!");
  }

  public void setMove(int x, int y)
  {
    this.move[0]=x;
    this.move[1]=y;
  }

  public int[] getMove()
  {
    return this.move;
  }
}

//
// Computer.java
//
public class Computer // Q1(d)
{
  private int[] move = new int[2];

  public Computer()
  {
    System.out.println("Computer Player created!");
  }

  public void setMove(int x, int y)
  {
    this.move[0]=x;
    this.move[1]=y;
  }

  public int[] getMove()
  {
    return this.move;
  }
}
```

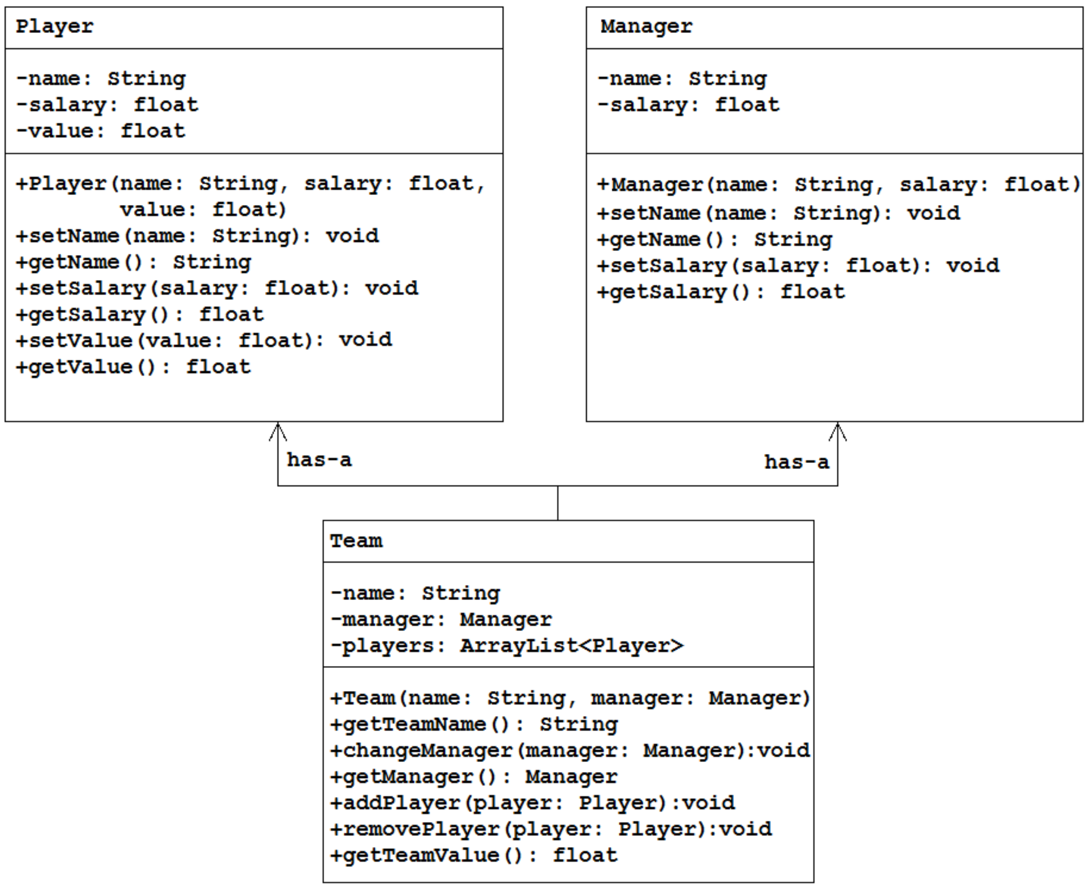
Appendix B

```
public class Manager
{
  private String name;
  private float salary;

  public Manager(String name, float salary)
  {
    this.name=name;
    this.salary=salary;
  }
}
```

Appendix C

(a)


Player
-name: String
-salary: float
-value: float
+Player(name: String, salary: float, value: float)
+setName(name: String): void
+getName(): String
+setSalary(salary: float): void
+getSalary(): float
+setValue(value: float): void
+getValue(): float
Manager
-name: String
-salary: float
+Manager(name: String, salary: float)
+setName(name: String): void
+getName(): String
+setSalary(salary: float): void
+getSalary(): float
has-a
has-a
Team
-name: String
-manager: Manager
-players: ArrayList<Player>
+Team(name: String, manager: Manager)
+getTeamName(): String
+changeManager(manager: Manager):void
+getManager(): Manager
+addPlayer(player: Player):void
+removePlayer(player: Player):void
+getTeamValue(): float

(b)

```
public class TeamTester {
 public static void main(String[] args){

  Team t1 = new Team("Man City", new Manager("P.Guardiola",5000000));

  Player p1 = new Player("Hart",1000000,5.0);
  Player p2 = new Player("Walker",1000000,6.0);
  Player p3 = new Player("De Bruyne",1000000,10.0);

  t1.addPlayer(p1);
  t1.addPlayer(p2);
  t1.addPlayer(p3);
  t1.removePlayer(p1);

  p1 = new Player("Ederson",1000000,6.0);
  t1.addPlayer(p1);

  System.out.println(t1.getTeamName());  // prints Man City
  System.out.println(t1.getTeamValue()); // prints 22.0

  t1.changeManager(new Manager("T.Devine",1000000));

 }
}
```

Appendix D

Sorting

```
public void selectionSort(int[] array)
{
  int temp; // temporary location for swap
  int max;  // index of maximum value in subarray
  for (int i=0;i<array.length;i++)
  {
    // find index of largest value in subarray
    max=indexOfLargestElement(array, array.length-i);
    // swap
    temp=array[max];
    array[max]=array[array.length-i-1];
    array[array.length-i-1]=temp;
  }
}

// Finds index of largest element
public int indexOfLargestElement(int[] array, int size)
{
  int index=0;
  for (int i=1;i<...;i++)
    if (...[i]>...[...])
      index=...;
  return ...;
}
```

Searching

```
int[] list = {26, 17, 5, 2}
int findValue=26;

if(sequentialSearch(list, findValue)==-1)
   System.out.println(findValue + " NOT found");
else
   System.out.println(findValue + " found");


// return -1 if not found, return index if found
public static int sequentialSearch(... , ...)
{
  . . .
  . . .
}
```