Homework Assignment 7

STEP 1 – Create Superclass and Subclasses

Create an Eclipse project named JohnDoeHw7, and use the default package, i.e., no package name should be provided. Then inside this project, create the parent class (superclass) **Player** as specified in the table below. Save it in a file ***Player.java***.

Also create the three child classes (subclasses) shown below. Save each class in a separate file – ***BaseballPlayer.java, FootballPlayer.java,***and***BasketballPlayer.java,*** also follow the template in ***CorrectSequence.java.***

Be sure to follow file ***CorrectSequence.java*** in Moodle folder “chap 4” zip file “UML-classDiagram.zip” to code Player and its subclasses, and this template file **includes the appropriate program header** and **comments in the class body**.

|  |  |  |  |
| --- | --- | --- | --- |
| **Class names** | | | |
| **Player**  ***(superclass)*** | **BaseballPlayer *(subclass of Player)*** | **FootballPlayer**  ***(subclass of Player)*** | **BasketballPlayer  *(subclass of Player)*** |
| **private data** | | | |
| id | number of hits | number of yards | number of shots made |
| player name | number of times at bat | number of rushes | number of shots attempted |
| team name |  |  |  |
| position |  |  |  |
| salary |  |  |  |
| commission rate |  |  |  |
| **public methods** | | | |
| constructors | constructors | constructors | constructors |
| getters | getters | getters | getters |
| setters | setters | setters | setters |
| calculate comission | calculate statistics | calculate statistics | calculate statistics |
| determine status | determine status | determine status |

**Detail specifications:**

In class **Player**:

* id is of int type; player name, team name, position is of String type; salary, commission rate is of double type.
* There should be two constructors, one is the default constructor with no parameter, and the other has all six parameters to initialize the six private data fields, respectively.
* There should be a public getter and setter for each of the six private data fields.
* There is one effector method to calculate the commission by multiplying the salary with the commission rate. This method return a double type.

In class **BaseballPlayer**:

* The two private data fields are int type.
* There should be two constructors, one is the default constructor with no parameter, and the other has all eight parameters to initialize the six inherited data fields plus the two private data fields, respectively.
* There should be a public getter and setter for each of the two private data fields.
* There are two effector methods:
  + **calculate statistics** : calculate the player’s batting average by dividing “the number of hits” by “the number of times at bats”, it should return a double type
  + **determine status** : boolean return type. If the batting average is more than **0.25**, return true; o/w return false

In class **FootballPlayer**:

* The two private data fields are int type.
* There should be two constructors, one is the default constructor with no parameter, and the other has all eight parameters to initialize the six inherited data fields plus the two private data fields, respectively.
* There should be a public getter and setter for each of the two private data fields.
* There are two effector methods:
  + **calculate statistics :** Calculate the player’s rushing average by dividing “the number of yards” by “the number of rushes” , it should return a double type
  + **determine status** : boolean return type. If the rushing average is more than **3.5**, return true; o/w return false

In class **BasketballPlayer**:

* The two private data fields are int type.
* There should be two constructors, one is the default constructor with no parameter, and the other has all eight parameters to initialize the six inherited data fields plus the two private data fields, respectively.
* There should be a public getter and setter for each of the two private data fields.
* There are two effector methods:
  + **calculate statistics :** Calculate the player’s shot percentage by dividing “the number of shots made” by “the number of shots attempted” , it should return a double type
  + **determine status :** boolean return type.If the shot percentage is more than **0.32**, return true; o/w return false

The three constant values mentioned in the determine status method as thresholds should be defined as public static final variable in the three subclasses, respectively. Refer to Orange.java in file “inheritanceExample.zip”, and see how the public static final variable is defined.

**DO NOT ADD or DELETE** any method from the requirements above, and you should strictly follow the requriements.

STEP 2 – Create UML class diagram for the Player class family

Follow document “DownloadAndUseVisualParadigmUML.docx” in file “UML-classDiagram.zip”, draw the UML class diagram for Player class and its three subclasses. Once it is finished, you need to export the UML class diagram as an jpg image file with name ***JohnDoePlayer.jpg****,* and replace JohnDoe with your first and last name.

As a reference, you can look at UML class diagram “Fruits.jpg” in file “inheritanceExample.zip”.

STEP 3 – Create an application class that uses the Player class family

In the same Eclipse project, create a new class file named *JohnDoeHw7.java*.

This application file has the main method, and inside the main method, it will create two instances of BaseballPlayer, two instances of FootballPlayer, and two instances of BasketballPlayer.

For each instance, you should use the constructor with 8 parameters, and then plug in all 8 parameters directly in the constructor. You can refer to source code “TestFruit.java” in file “inheritanceExample.zip”, and see how object myOrange is constructed.

You can use arbitrary values for the 8 parameters in each instance, as long as they are reasonable, for example, in basketball player, number of shots made should be less than number of shots attempted.

Then for each player instance, you need to output the six inherited data fields from Player class, and its commissions, then output the two private date fields, and then the statistics of this player, and the “keeping status” of this player: true of false. An example output for a football player is as below:

player id: 20

name: Barry Sanders

team: OSU

position: running back

salary: 1000000.00

commission rate: 0.02

commission: 20000.00

number of yards: 2850

number of rushes: 373

statistics: 7.6

keeping status: true

The output precision requirement: for monetary items, two digits after the decimal point; for statistics, one digit after the decimal point. Use ***System.out.printf()*** method to format the String type and boolean type (%s), double type(%f), and int type (%d) output items, with format specifier %s, %f, and %d, respectively. For double type, you need to specify the number of precisions after the decimal point.

Generate a simialr output as above for all six players. When you hardcode parameters to initialize the six playes, you can carefully design the parameters so that in each sport category, one player has “keeping status” as true, and the other player has it as false.

STEP 4 – CHECK YOUR WORK FOR ADHERENCE TO PROGRAM SPECIFICATIONS

#### NOTE – ADDITIONAL SPECIFICATIONS!!

#### You do NOT need to use any array in this homework.

#### The names of the classes must be exactly as stated in the table in page 1 of this document.

#### You can not CHANGE, ADD or DELETE any instance variable or method to the Player class and its subclasses. However, inside each method, you have the freedom to use and name any local variable that you need. Also in the main method of the application class, you have the freedom to name and use any local variable.

#### For class Player and its subclasses, you must provide a constructor without arguments and a constructor with a full list of arguments to initialize all inherited data fields (if applicable), and all private data fields.

#### In main method, you must use constructors with actual parameters to create the required six objects, by hardcoding the arguments in the contructors. NO USER INPUT is needed in this homework!

#### For the determine statistics method in each subclass, the division should happen between two int type variables, and the result should be double. How do you achieve that? Hint: in the division statement, type converting the dividend from int to double will yiled a double result, such as (double) dividend / divisor

#### Except for the three threshold values in the three subclasses, all class data must be private, and you must provide public getters and setters for each private instance data.

#### The methods to calculate statistics must have the exactly SAME name in all classes.

#### The methods to determine the player’s status must have exactly the SAME name in all classes and must all return a boolean.

#### You do NOT need to use any abstract class or abstract method in this homework.

#### Put each class in a separate file –*Player.java, BasketballPlayer.java, FootballPlayer.java,.java*, *BaseballPlayer.java,* and *JohnDoeHw7.java*(for application program that has the main method).

**What and where to submit:** you need to ***zip*** these files together, and name the zip file ***JohnDoeHw7.zip***:

* *Player.java*
* *BaseballPlayer.java*
* *FootballPlayer.java*
* *BasketballPlayer.java*
* *JohnDoeHw7.java*
* *JohnDoePlayer.jpg*(this is the exported image file of the UML class diagram for class Player and its subclasses)

Submit this zip file to Moodle homework 7 drop box. After submission, be sure to follow the **VERIFICATION** process: download your .zip file to a local folder in your computer, unzip the zip file and then create a new Elcipse project, and include the extracted java files into the project, and compile then run the Eclipse project. If the verification process works correctly, then you have submitted the .zip file successfully in Moodle; if not, you need to fix the problem and re-submit. **DO NOT zip any eclipse project folder into your submission zip file**. If the submitted java files program **does not compile or does not run, your score will be zero**. **If the user inputs are all correct, but your program still has a runtime exception,then it will also result in zero point.**

For how to locate the java source code files, and how to zip multiple files into one zip file, please refer to “chap1-schedule.docx” for instructions.

This assignment has **20 points. Grading components is listed in the table below:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Player class and its subclasses** | **main method implemented correctly** | **Correct UMLclass diagram** | **Code indent & align** | **program header,**  **comment, naming convention** | **Submit file format** | **Program logic** |
| **8** | **5** | **3** | **1** | **1** | **1** | **1** |

The grading components are, but not limited to: code alignment and indentation, variable/method/class naming convention, programmer header, suitable comments, submitted file format, overall program logic.

For Eclipse java file, if there is still any red circle check mark in java source code, you will receive zero point, because red circle check mark in Eclipse IDE means the java source code still has compilation error. You have to fix this compilation error first, before you proceed to the next phase of running java program. Even in some rare scenarios, you can run the program in Eclipse with compilation error, but a compilation error in java source code will result in zero point for the homework.

When coding in Eclipse or Greenfoot, please read document “RulesForIndentAndAlignCode.docx” in Moodle folder “chap 1”, and follow all the rules in code alignment and indentation.