|  |  |
| --- | --- |
| **Class Diagram**  **All questions in this exam are based on one app.** |  |
| **Skeleton Driver Code** | | |
| **Skeleton Gym Code** | | |
| **Skeleton Gym Member Code** | | |

**Programming Fundamentals 1 – in-class test – Sample Test**

|  |  |  |
| --- | --- | --- |
| **Name** | **Student Number** | **Course** |
|  |  |  |

|  |
| --- |
| **Instructions:**   * **1.5 hour exam.** * **3 Questions, answer all.** * **Fill in your name, student number and course above.** * **Complete the areas in this booklet and give it to your invigilator before leaving the room.** * **Submit the section with your code as a complete booklet.** * **You can write anywhere in this booklet and use the back of pages for additional code, rough work, etc.** |

With this exam, you are given class diagram

for the classes:

* **Driver**
* **Gym**
* **GymMember**

Which make up a Gym System.

You will be asked to write code for the following classes:

* GymMember
* Gym
* Driver

**Question 1 – Gym Member Class:**

Note: you do not have to define the fields or write the getters.

|  |
| --- |
| GymMember.java |
| //Code the the constructor, the setters and the toString() methods as defined in the comment in the following boxes.  // You should implement the following validation rules :  // For the **member name field**, this should be a maximum of 20 characters.  // In the case of the constructor, names with more than 20 characters  // should be cut to the first 20 characters.  //(Hint : You can use the String method ***substring*** to help you)  // In the case of the setter, do not update the value if it is more than //20 chars long.  // For the **membership number field**, the valid values are between 100 and 999 //(inclusive) . In the case of the constructor, if an invalid value is  // input, the default value of 999 should be used.  // In the case of the setter, do not update the value if it is outside  // the valid values.  // There is no validation needed for the **isCurrentGymMember, height or weight fields**.  1.1 Setters :  public void setName(String name) {// **FILL IN CODE BELOW**  }  public void setMembershipNumber(int membershipNumber)  {// **FILL IN CODE BELOW**  }  public void setCurrentGymMember(boolean currentGymMember)  {// **FILL IN CODE BELOW**  }  1.2 Constructor:  public GymMember(String name, double height, double weight, int membershipNumber, boolean isCurrentGymMember) {  //FILL IN THE CODE INCLUDING IMPLEMENTING VALIDATION RULES  }  1.3 toString()  public String toString() { // FILL IN CODE THAT RETURNS A STRING //VERSION OF THE OBJECT  } |

2. The class diagram has a Gym class. In this class, there is an array of member objects, defined as follows:

private GymMember[] members;  
int total = 0; *//number of members that have been added to array*

|  |
| --- |
| Gym.java |
| // Code each method as defined in the comment in the following boxes.  2.1 – add(..)  // If there is space available, add the GymMember object, passed as a parameter, to the array. // parameter is gymMember which is a GymMember object to be added to the array. // returns the Status of the add; true for success, false for fail. public boolean add(GymMember gymMember) { //FILL IN CODE BELOW  }  2.2 listGymMembers()  //The return type is String. // This method returns a list of the gym members stored in the array  // list. Each member should be on a new line and should be  // preceded by the index number e.g. // 0: Member A ……… // 1: Member B ……  // If there are no members stored in the array list, return a  // string that contains "No Members in the Gym".  public String listGymMembers () { // FILL IN CODE BELOW  }  2.3 listGymMembersOver56kg()  *// This method builds and returns a String containing all the*  *// members in the array // whose weight is over 56*  *// returns a String containing all the members in the array whose*  *// whose weight is over 56or // "No members are heavier than 56kg", // if none in the array. // If there are no members in the array, the returned String*  *// contains "No Members in the Gym"* public String listGymMembersOver56kg () { FILL IN CODE BELOW      }  2.4 getGymMembersTallestMember()  */\*\* This method goes through the array of members and returns the tallest member*  *\* . If no members exist in the array, null should be returned.  \*/* public GymMember getGymMembersTallestMember (){//FILL IN CODE BELOW  } |

**Question 3: Driver**:

The class diagram has a Driver class. In this class, there is an object of the Gym class.

**public class** Driver {  
  
  
private Scanner input = new Scanner(System.*in*);  
private Gym theGym;  
  
public static void main(String[] arg) {  
 new Driver();  
}  
  
public Driver() {  
 runMenu();  
}

|  |  |
| --- | --- |
| This class displays the menu of options that the user can choose from: |  |
| This class uses the Scanner class for reading from the console: |  |

3.1 addGymMember()

In the following box, complete the code for the “Add a Gym Member” menu option. The add method that you will be calling in Gym has the following header:

private void addGymMember()

|  |
| --- |
| Driver.java |
| **private void** addGymMember(){  **// TODO Write the code to read in the data for a**  **// GymMember and add it to the Gym.**        } |