Assignment brief

Time 3 hr (+1 hr for upload)

You have been given some code that is currently being developed as part of a Computer Game **Sales Analysis System**. The system will allow analysis of available sales data across a number of different international markets and other categories.
Not all the requirements have been implemented. It is your task to implement these and raise the coding standards of all the code.

Create a **project solution** (named **<Your Name>_<Student Number>_p3 e.g. MatthewCollins_12345678_P3**). Create a **package** named **p3Aug**.

2 initial files have been provided.
Add **StartApp.java** to the p3Aug package in the solution and add **videogamesalesdata.csv** file at the project root.
Ensure your name and student number are placed in the Javadoc comments of all the classes you create.

- The StartApp has been partially completed with a menu and a partially completed readData method.

The application will run (start) from the StartApp.java, initially reading in the data from the **videogamesalesdata.csv** file and then perform a number of menu driven operations.

Assessment Structure

3 Parts. Weighted 30% each with the remaining 10% for general approach and style.

Part 1: Implement the **Game** Class based on the structure of the data in the csv file and the specification details given in this document.

Part 2: Appropriately Unit Test the **Game** Class. Further details later in this document.

Part 3: Edit and further develop **StartApp.java** to read the data from the file into a JCF List of Game objects and implement the missing menu driven functionality.

Part 1 – Game class implementation

Using your knowledge of OOP you should add/update code based on the following:

Analyse the data in the **videogamesalesdata.csv** and create a class (**Game.java**) and any other associated files/classes you deem necessary for your design.

Each column in the file corresponds directly with an expected attribute of the object.
The following allowable values/business rules have been agreed.

Attribute	Allowable Values
Name	At least 1 character but no more than 80
Platform	Any of the following values (abbreviations agreed as per file contents): 3DS, PC, PS3, PS4, PSV, WiiU, X360, XOne, Any
Release Year	Any Year from 1980 onwards
Genre	Any of the following values Platform, Role-Playing, Shooter, Sports
Publisher	At least 1 character but no more than 40
North American Sales (units – millions)	Non-negative number
European Union Sales (units – millions)	Non-negative number
Japanese Market Sales (units – millions)	Non-negative number
Other Markets Sales (units – millions)	Non-negative number

Appropriate getters/setters should be provided for all attributes.

An attempt to set a value outside of allowable business rules should give an appropriate exception with an appropriate exception message e.g. “Invalid Publisher name” etc.

The implementation of the overall system also assumes that the Game class will have the following methods which you will need to implement.

getGlobalSales()	This method should return total Sales numbers across all regions
getBestMarket()	Dependant on which of the 4 markets has the highest overall sales numbers for the current object this method should return the name of that market. It has been agreed that this should be returned as text. Expected return values: • North America • Europe • Japan • Other

Other methods etc. can be added as the developer deems necessary.

Part 2 – Unit Testing Expectation

Conduct appropriate Unit Testing of the behaviour of the **Game** Class and its methods.

For the purposes of this assessment, it is not necessary to test any other classes/methods.

It is recognised that there is some repetition in the requirements of some of the fields, mainly the 4 sales data fields.

Therefore in the interest of time saving **it will be acceptable to fully and rigorously test the functionality just one of those sales fields.** For the other fields simple verification of the getter/setter setting values will be enough. (in the real world this should be fully tested but can be skipped in this time limited scenario without penalty)

It is not necessary to test void methods which have no impact on the state of the object. (e.g. printDetails type methods)

If fully autogenerated methods such as equals() are deemed necessary and included in your implementation it will not be expected that unit tests of these are included for the purposes of the assessment.

Part 3 – StartApp and Menu Options

1) The **readData** method has been partially completed. Finish the implementation of this method so that it will work with your **Game** class. (readData method puts game values in a variable called **gameSalesData** which will then be used by other methods in the program).

2) Display the total **number of Games** in the current List (repeatable as may change due to other menu options)
   Sample output:
   ```
   Number of games in current list: 311
   ```

3) Display **full details** for all games in the current list.

   Sample output:

```
name:           Fallout 4
platform:       PS4
release:        2015
genre:          Shooter
publisher:      Bethesda Softworks
Sales
North America:  2.45 million
Europe:         1.26 million
Japan:          0.03 million
Other:          0.69 million
Total Sales:    4.09 million

Etc.
```

4) Display **summary details** (name and key info) of the **top 3 Best Selling Games in the European Region** in the current list

Sample output (not necessarily the top values in the actual file):

```
1)
FIFA 16 – PS4 (2015)
NA: 1.11 | EU: 6.06 | JP: 0.06 | OTH: 1.26 | Total: 8.49

2)
Call of Duty: Black Ops 3 – PS4 (2015)
NA: 5.77 | EU: 5.81 | JP: 0.35 | OTH: 2.31 | Total: 14.24

3)
Etc. etc.
```

5) Display summary details of **Platform Games** in the current list with **total global sales of 0.2 million or more**, sorted **in descending order by total sales** (same layout as other methods using summary details)

6) Display the **name, genre and publisher** of the game/games with the **longest name** in the current list

7) Find Multiplatform Games. Some games e.g. FIFA 16 have been produced for multiple platforms (PC, PS4 etc). Identify and **display summary details of any such games** in the current list. **Sorted alphabetically by name**

8) Identify any games in the current list which **sold only in the Japanese market**. Display **summary details of all such games in descending order by Japanese sales numbers**

9) **Remove/Combine any duplicate games from the current list**. (The effect of this can be verified with other menu options eg. Option 2 or 7. And can be reversed by reading the file again with Option 1)
The currently provided input data file separates games by platform. But the platform field in the spec also has an allowable value of "any". **Update the current list so that all game objects have a platform value of "any", each game only appears once, and relevant sales data for games which originally occurred more than once are added together.**

Complete as much as you can within the time available balancing your efforts across the 3 sections. When finished, compress (zip) the entire **Eclipse solution** and upload to Assignments (P3 August Retake assessment) on CANVAS

Now : check the uploads to ensure you have submitted the correct files
[END]