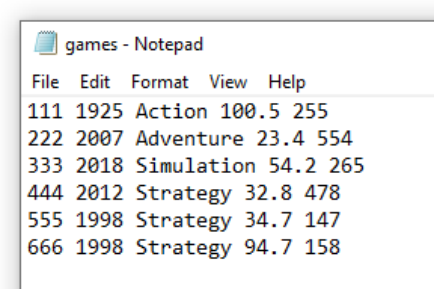
Department of Computer Technology and Information Systems
CTIS221 – Object Oriented Programming
Fall 2019 - 2020
Lab Guide 17 - Week 12-2

OBJECTIVES: Collections and Text Files with GUI
Instructor : Burcu LİMAN **Assistants**: Burcu ALPER, Leyla SEZER

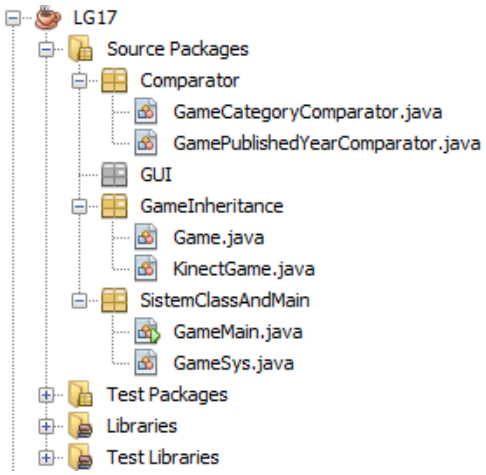
Write a Java program that reads game information from a file named **"games.txt"**. In part b, program creates a GUI for the related information first gets the data from the file, then makes add, display and search operations according to the type of the game**.**

PART A: <u>Implement your classes</u>
Your program will get input from 1 text file with the following structures;

The **games.txt** file includes information about games as; game id, published year, game category, burned calorie and point.



You are going to create each class in separate java files. Packages and the file names are as follows;

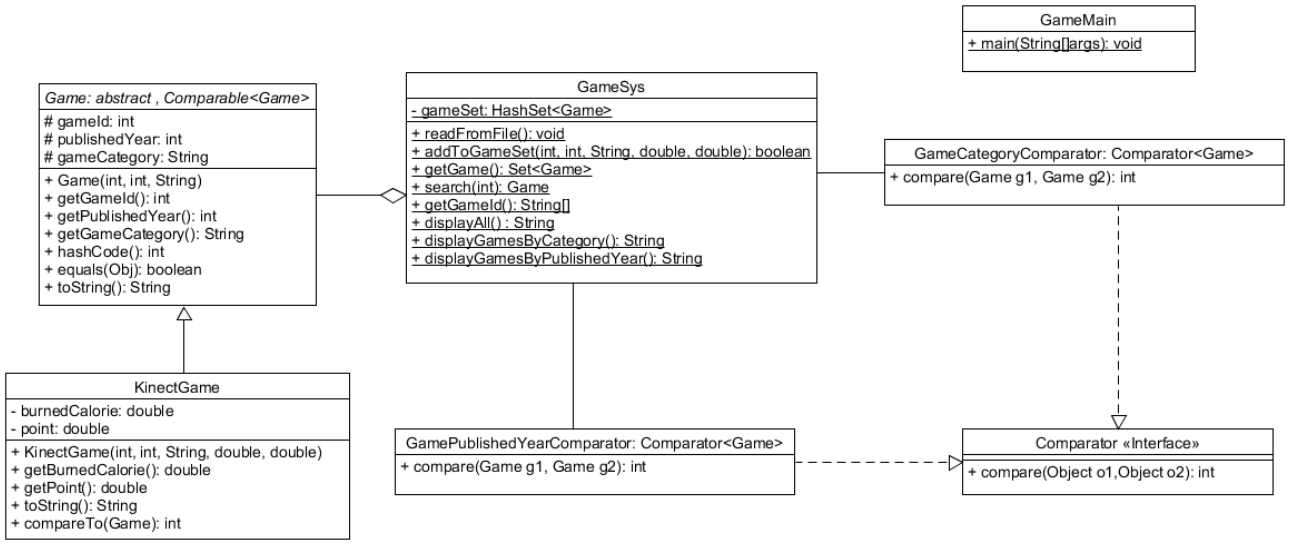


➔ Check the UML class diagram, implement your classes according to it, and do not change visibility modifiers.

➔ Write only the necessary accessor and mutator methods inside the classes!! You may add methods other than the given ones, if you are going to use them!!

➔ toString() method for all the classes will return the necessary data field information **<u>related with</u>** the implemented class.

➔ There is a HAS-A relationship between;
- Game - GameSys



Information of the Game class structure:

- Implement the given data members and methods according to the uml-class diagram.
- Write a **hashCode()** and **equals(..)** methods for the game id.

Information of the KinectGame class structure:

- Write a **compareTo(..)** method that will take a Game object as a parameter and compares the game ids in the ascending order.

Information of the GameCategoryComparator class structure:

- Define the method **compare**() that takes two Game objects as parameters and compares two object's category.
  Hint: There can be more than one object with the same category.

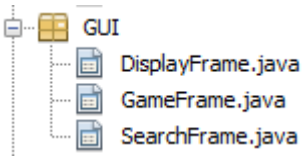
Information of the GamePublishedYearComparator class structure:

- Define the method **compare**() that takes two Game objects as parameters and compares two object's publishedYear.
  Hint: There can be more than one object with the same published year.

Information of the GameSys class structure:

- Create the data member***:***
  - A hashSet which stores ***Game*** objects.
- Implement a **readFromFile()** method, which reads information from the text file and stores them in the hashSet.
- Implement a **addToGameSet(…)** method that takes the information of the Game. It creates a Game object and add to the gameSet. If the game object is already exist return 0, otherwise return 1.
- Implement a **getGame()** method that returns the gameSet.
- Implement a **search(…)** method that takes a gameId as an input and searches a game object with the gameId inside of the gameSet. If it founds, returns the object otherwise returns null.
- Implement a **getGameId()** method that returns the id's of the Game objects in the gameSet as a String array in sorted order.
- Implement a **displayAll()** method that returns the content of the hashSet.
- Implement a **displayGamesByCategory()** method that returns the content of the gameSet in ascending order according to the game category by using the GameCategoryComparator class.
- Implement a **displayGamesByPublishedYear()** method that returns the content of the gameSet in descending order according to the published year of the game by using the GamePublishedYearComparator class.

PART B: Implement your GUI



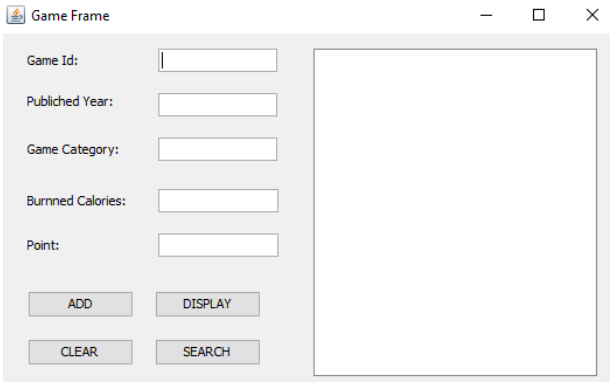
1) When program starts in main class;

a. File will be read by invoking the **readFromFile()** method.

b. The start-up frame should be created and sets its visibility to true;

i. There are 3 buttons, 5 labels and 5 text fields as shown below.

ii. Set the title to the "Game Frame".



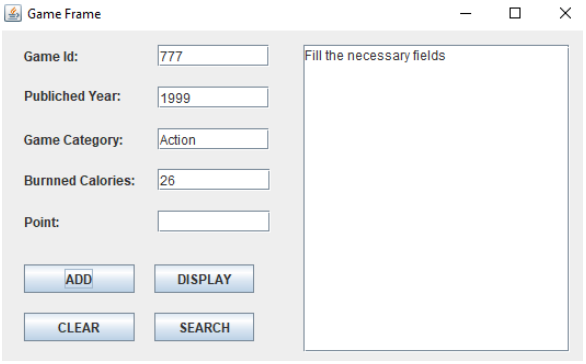
c. When the user clicks on the "ADD" button,

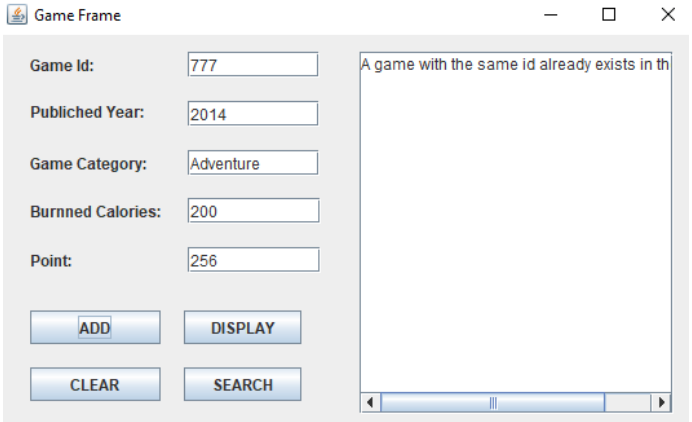
i. Create a game object and add to the gameSet by invoking the **addToGameSet(..)** method.

ii. All ids of the game objects in the gameSet will be shown on the combo box as sorted in ascending order in the SearchFrame.

iii. Display an appropriate message when;

- Addition is successful or not
- There is any empty text field

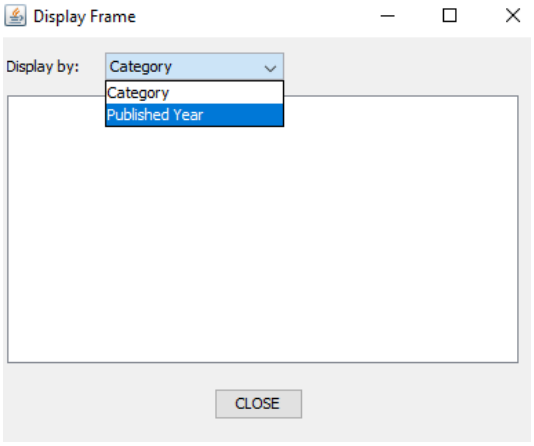




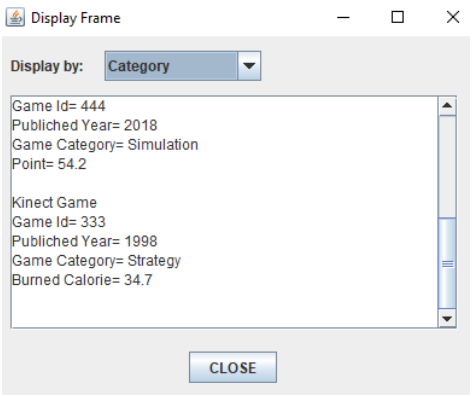
d. When the user clicks on the "CLEAR" button clear all the text fields and the text area.

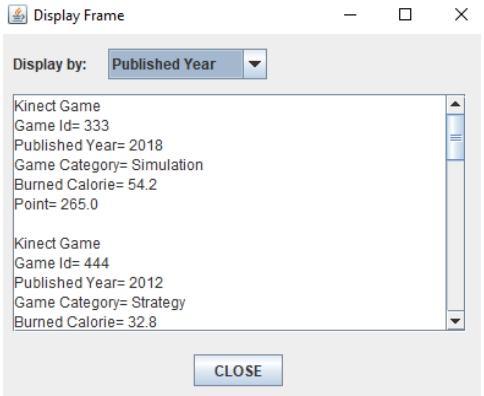
e. When the user clicks on the “DISPLAY” button;

i. If the hash set is empty, give a warning message and do not open a frame
ii. Otherwise; A frame should be created and sets its visibility to true;
iii. Set the title to “Display Frame”
iv. There are 1 buttons, 1 combo box, 1 text area, 1 labels as shown below.
v. Display the content of the gameSet in the text area as sorted according to the game category.



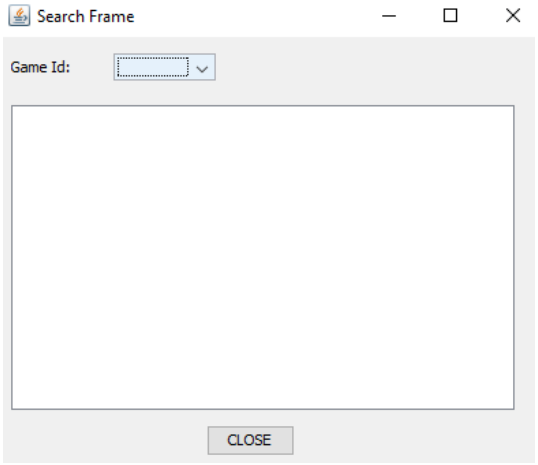
- When “CLOSE” button is clicked dispose the frame.





f. When the user clicks on the “SEARCH” button;

i. If the hash set is empty, give a warning message and do not open a frame
ii. Otherwise; A frame should be created and sets its visibility to true;
iii. Set the title to the “Search Frame”
iv. There are 1 buttons, 1 combo box, 1 text area, 1 labels as shown below.
v. All ids of the game objects in the gameSet will be shown on the combo box as sorted in ascending order.



- When an id is selected from the combo box, display the content of the game with that id on the text area.
- When “CLOSE” button is clicked dispose the frame.