Module
Back to the classics!

1. Objective

A well-known video game store company wants to reward its customers by offering them the chance to play a game based on the aesthetics of the characters from the (wonderful) console games of the 80s (you can go back in time and meet the great-grandfathers of games at https://www.classicgamesarcade.com/).

For this purpose, several terminals have been installed in each store from which each customer can play a round of the game described below and, if they are lucky, win some awards.

2. Goal

We have to design a visual application that implements a strategy game to try to avoid the invasion of our territory (a game board) by some characters from video games of the 80s. The application will allow a single user to play and the opponent will be the computer, which will randomly generate five by five characters ready for the invasion. The user's goal will be to avoid the total occupation of the board trying to group the characters so that they disappear from it, according to the rules described below.

In addition, depending on the final score obtained in the game, the user will be able to choose from the same terminal the prizes to which they can choose.

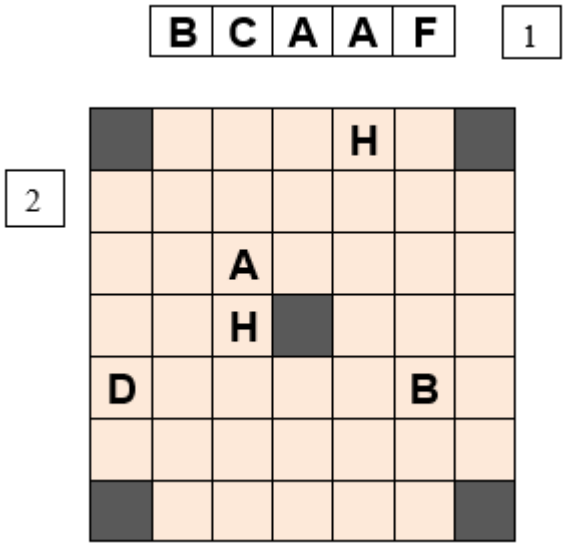
3. Basic rules and game development

3.1. Territory and Invaders

- The game takes place on a board with 7 rows and 7 columns. All the squares can be occupied by a character (or invader) except for the squares in the corners and center of the board, which will be considered non-accessible positions in the territory.
- In the game, eight different types of characters from classic video games will be considered, at the student's choice. One of them (the one chosen by the student) will be the leader of the invasion and, therefore, will have an important role in the game, as will be described later.
- At the start of a game, 5 randomly generated invaders will have already occupied part of the territory or game board, in randomly generated positions. The rest will start with their invasion

strategy appearing randomly 5 at a time over 10 successive iterations. In each of them, the invaders wait for the user to assign them the part of the territory (one of the squares on the board) that the user decides.

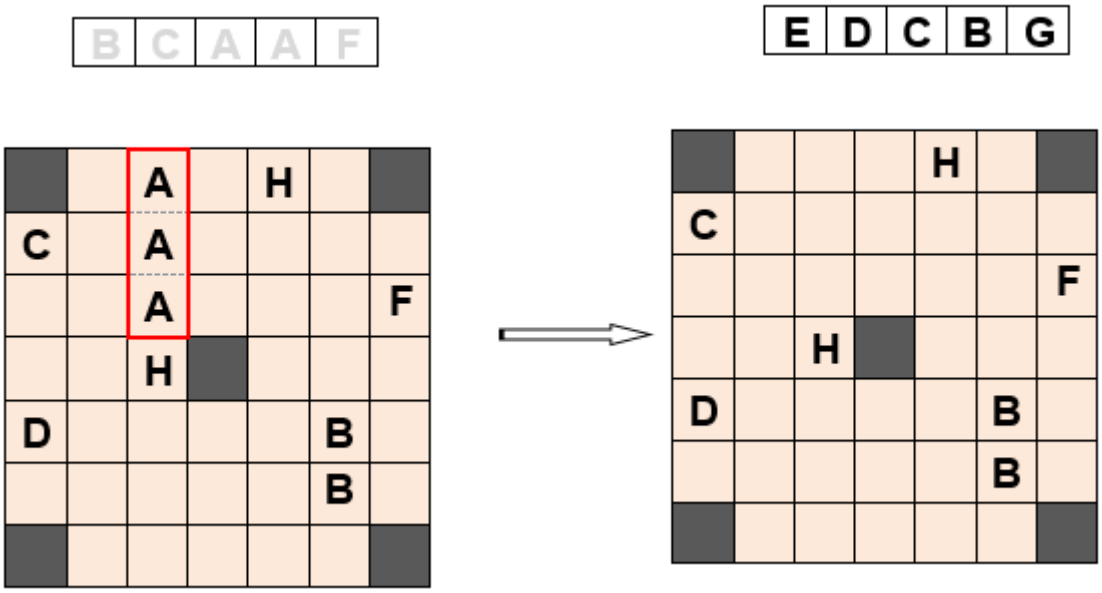
- Example: We consider the following 8 types of characters: A,B,C,D,E,F,G,H. The initial situation of a game could be the following, being (1) the set of 5 characters waiting to occupy a cell of the board and (2) the state of the board at the beginning of the game, with 5 characters initially located in different cells of the board:



3.2. Turn development

- The player has to place each of the 5 invading characters on different cells of the board, placing them in the order and positions that they want, bearing in mind that a cell that is already invaded cannot be occupied by a new invader. **In each turn**, the player will try to form groups (or colonies) of the same type of invader of at least 3 individuals, since once a colony of 3 or more individuals of the same type is formed, it will disappear from the board **at the end of the turn**[1]. Characters can be grouped in the same row or column: characters located in adjacent cells on the same diagonal will not form colonies.
- Example: Starting from the starting situation above:

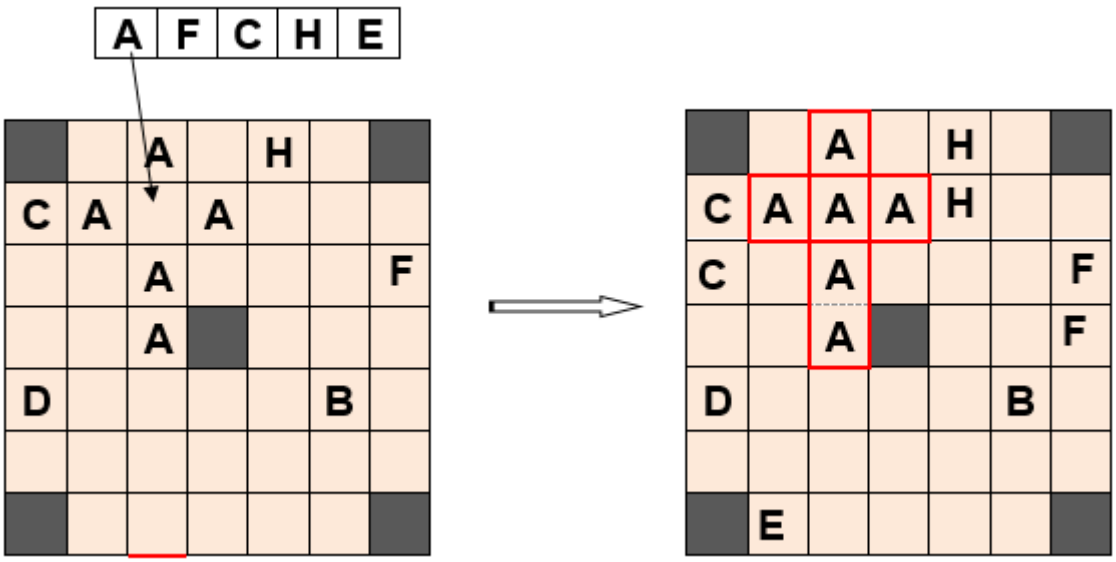
[1] That is, the system will not evaluate the colony until the 5 invaders have been placed.



- Once the 5 characters have been placed in different squares and the situation on the board has been updated, another 5 will appear (turn 2) that must be placed, and so on until the end of the game.

3.3. Scoring

- The player starts from an initial balance of 0 points. Current score must be always visible and updated in the interface. This balance will increase in each interaction by 50 points when making a group of 3 invaders disappear, by 200 if the group is of 4, 1000 if it is of 5, 5000 if it is of 6 and 10000 if it is of 7. If more than one colony is destroyed in the same iteration, the corresponding points of each of them will be added. Example:



3.4. Game over

The game finishes in one of the following situations:

- When **a colony of 5 or more leader type** characters. It is considered that if 5 or more of these individuals disappear at once, the invasion ends cause a large part of its leaders are eliminated. In this case, no more invaders will arrive (so we will not reach the 10th turn), the user is the winner and 20000 points are added to their final score.
- **The last turn (the 10th)** is reached and the board has some free space. In this case, the user also wins since they avoided the total invasion of the territory and ends the game with the accumulated score.
- When **all the squares on the board are occupied by an invader** and there is no possibility, therefore, of continuing to place new invaders. In this case, the invader is considered the winner and the user ends with 0 points.

4. Functional Requirements

- When starting the application, the code and name of the store that corresponds to the terminal where the game has been installed will be read from a **config.dat** configuration file. The store code will only be used for ticket validation purposes, as described below, so it will not be displayed to the user in the application. The format of this file is as follows:

```
store_code@store_name
```

Example:

```
33429_01@VideoGame Parque Principado
```

- All customers of the video game store who have made a purchase for an amount greater than or equal to 20 euros will have the right to play a game (Note: each ticket, whatever the amount, as long as it is equal to or greater than 20 euros, will give the right to ONE SINGLE round. A ticket of, for example, 45 euros, will not give the right to two rounds, just one).
- In order to validate the tickets, it is assumed that each ticket generated in the store is stored in real time in a **tickets.dat** file that must be read at the beginning of the application. This file stores the number of each ticket and its amount. The ticket number is formed by the store code concatenated with a 5 digits number.

The format of said file is as follows:

ticket_number@amount

Example:

```
33429_0112314@10
33429_0124672@110.75
33429_0136517@25.85
```

Note: The insertion of the data in said file is not in the scope of this module of practices. There should be no verification of the different codes of the file or the format of the amounts; they are all considered to be correct.

The option of constantly reading the file with the updated data **should not be considered** eitherreading it when starting the application is enough. However, it must be updated by eliminating the tickets that have been already used to play.

- The user, who has a copy of the purchase receipt, will enter the ticket number in the visual application. The application will validate that the data entered corresponds to a ticket that meets the requirements to qualify for the game:
  - Corresponds to the store where the terminal is installed (ie, the store code corresponds to the one in the config.dat file).
  - The amount is equal to or greater than 20 euros.

If so, the user will be able to start playing. Otherwise, if with that ticket they do not have the right to play, they will be informed in the manner deemed appropriate.

- If the user ends the game with more than 0 points, a selection of prizes will be presented for the user to choose from. The list of prizes is in a file **awards.dat** with the following format on each line:

```
award code@name@description@type@value in points
```

Being:

- **Award code**: Sequence of a letter and two digits that uniquely identifies each prize. It will be treated as internal data and will not be shown to the user in any case.
- **Name**: Name of the award
- **Description**: Award details
- **Type**: Category to which the gift belongs. We consider only the following: Accessories, Consoles, Videogames
- **Value in points**: points for which the prize is exchanged.

Example:

```
A01@Turtle Beach Headset@Immerse yourself in your games with the powerful amplified sound and bass boost of the Recon 200@Accessories@150
A02@Pack of 2 steering wheels@Transform your Joy-Con controllers into steering wheels and drive taking advantage of motion control technology in games compatible with the set of two steering wheels for Nintendo Switch@Accessories@50
A03@Razer Basilisk V3@ MouseDefend your gaming style with the new ergonomic mouse par excellence. With 10 + 1 programmable buttons, a smart scroll wheel@Accessories@200
C01@CONSOLA RETRO MY ARCADE PAC-MAN@Contains four classic Pac-Man titles: Pac-Man, Pac-Man 2, Pac-Mania and Pac-Panic@Consoles@1050
C02@Playstation 4 500Gb + NBA 2k21@The best selling console in the world, now with a new look and also with amazing HDR graphics. Enjoy incredibly vivid and brilliant colors. Includes the sports simulator par excellence, NBA 2K21, to live your NBA experience at the next level@Consoles@8000
V01@Battletoads & Double Dragon NES@8-Bit retro game, where two players can choose between five fighters and venture through treacherous settings using all possible techniques to kill enemies@Video Games@300
V02@Farming simulator 15@Manage and grow your farm: plant crops, raise livestock (cows, chickens and sheep), sell fresh produce...@Videogames@300
V03@FIFA23 PC@EA SPORTS FIFA 23 brings us even closer to the action and realism of the world game thanks to HyperMotion 2 technology, with twice the real-world motion
```

```
capture and animations that are more authentic than ever in every match@
Videogames@500
```

In addition, each of the gifts will have a photograph associated with it, the name of which will be the code of the gift and the extension ".png". These images may be of any size, so the application must dynamically adapt them to the size of the component that shows them.

- Users will choose the prize or prizes he wishes, as long as the sum of the points of their selection does not exceed the score obtained in the game. They may choose more than one unit of the same item, if they wish.
- The user must be able to filter the gifts by categories so that they can see the list of prizes
- The chosen prizes and the remaining points must be visible on the interface.
- As long as the user does not definitively confirm the choice of their prize/s, they can make the changes they want.
- To formalize the choice of prizes, the user will enter their ID or identification document (do not validate any format). At that time, the application will notify the user that they can pick them up whenever they want.
- The selection of prizes made by the user will be stored in a file **deliveries.dat** that will be saved in the system so that the user can claim the prizes in the store. The format of this file is as follows:

```
user_national_id@store_code@prize_code1@prize_code2...
```

- When the current user finishes their game (including the possible selection of prizes), the application must be fully prepared for interaction with a new client.
- Once the user has accessed the application using a ticket, the ticket will no longer be valid to play again.
- A menu will be included with the options that the student deems convenient. The About will necessarily include the **student's name, ID and lab group**.
- The application must incorporate a complete help system with at least 5 html files.

5. Non Functional Requirements

- The application must be developed and be compilable and executable with the eclipse and java versions used in the development of the subject labs.
- All the files (input and output) must be located in a **files** folder in the root directory of the project, as was done in the practices.
- All the images necessary for the application must be located in the **src/img** folder, not including any subfolders.
- The screen that contains the most relevant elements of the application (the one that contains at least the dashboard) must be resizable.
- The interface will have to respect the original design made by the student and refined after the evaluation process carried out (Wireframe V2). **Any change regarding said design will have to be duly justified in the documentation.**
- The application will have to be prepared to adequately manage any number of elements contained in the files. It will not be necessary to consider possible changes in the format of the files or the existence of erroneous values in them. **The modification of the file formats is not allowed**; this would mean that the module would not work with the files used in the correction and therefore, the **module would not pass**.

6. Optional Extensions

- Internationalize and localize the application for Spanish and English.

7. Development and UI Aspects. Evaluation

- **Remember that a clear separation must be established between the code corresponding to the visual presentation and that which represents the business logic of the application**; Serious errors in this aspect will not be admitted to reach the minimum mark in the module. A package structure similar to the one developed in the subject practices must be implemented.
- In order to facilitate testing, a DEBUG constant must be included in the Main class that contains the main method of the application, which will determine the behavior of the game in such a way that:
  - If DEBUG is 0, it will work according to the rules described above.
  - If DEBUG is 1, the 5 aliens that spawn each turn will no longer be random and will all be "leader" type
  - If DEBUG is 2, all 5 aliens will be equal but any type other than the leader.
- Given the nature of the subject, the graphic interface must be consistent with the guidelines given in theory, applying the rules and recommendations regarding the design of user interfaces. The adequacy of the selected components in the development of the application will be especially valued.
- All components used in practice shall have a meaningful name. The use of names like jNewButton will be penalized.

8. Documentation

In addition to the files that are part of the project, an explanatory document must be delivered with the solution adopted to solve the proposed problem. This document will have the name "PL-I-XX-Name-Surname-DNI" of the student, replacing the XX for the number of the lab group, and must contain, at least:

- **Cover with student identification**
- **Table of Contents**
- **Introduction**: the problem to be solved will be presented. It will be clearly indicated if the option to raise grade (internationalization) and any other element that has to be taken into account when reviewing (use of sound, for example) has been included.
- **Application development:**
  - **Design phase**:
    - The screens with the design made by the student and delivered as Wireframe V2 **must be included in the documentation**. If any change has been made in the final design with respect to this wireframe, it must be reasonably justified.
  - **Implementation phase**
    - **Logic**: description and explanation of the class(es) added to solve the logic part of the application as well as the most relevant methods of each one of them. Clearly indicate which method generates invaders of the same type and at what point in the code it must be called to generate all invaders of the same type.
    - **Interface**: Snapshot of the screen/s (windows and/or panels) that make up the application with a relationship and justification about the choice of the most relevant components of each one.

- **Tests**: The results obtained after carrying out the 7 scenarios used to validate the wireframe must be documented once the development of the module has been completed. If an error has been detected that has implied its correction, the scenario, the error found and the solution implemented must be documented.

9. Delivery Rules

A compressed file with the name of the student's ID will be delivered. Upon unzipping the file, the teacher should get:

- The explanatory document described in the previous section.

- The complete folder of the project (do not include workspace). The project will have the name **Mod-PL-I-XX-Name-Surname1-ID**, where PL-I-XX is the student's lab group.

The delivery will be made through the Virtual Campus. The deadline for delivery will be Sunday, **January 8 at 9:00 p.m**. **Deliveries after that date or deliveries made by email will not be accepted.**

10. Others

The realization of this practice **is strictly individual**. Each student is responsible for preventing their practice from being copied or plagiarized in any way. Cheating in practice (as in any other exam) is a very serious action, and teachers reserve the right to adopt appropriate measures.

The exam on the module will be held on the day established for the ordinary exam in January (Thursday **12/01/2023**, 3:00 p.m.). If multiple slots needed to be established, they will be mailed the day before.

Regardless of this exam, if the teacher deems it appropriate, a student may be summoned for individual defense of their practice module.