OBJECT ORIENTED PROGRAMMING PROJECTS 2020 - 2021

Contents

Common Requirements

- Your project must implement all OOP concepts: Encapsulation, Abstraction, Inheritance and Polymorphism.
- You will be mainly evaluated on how you used the OOP concept in your idea.
- Any other libraries or engines, you will use, will be considered and evaluated as a helper item and it should be written in Java.
- Your program must be divided into PROCs, each of which is responsible for one and only one functionality
- Keep your code clean, follow a specific convention, and choose meaningful identifiers (variables and methods names).
- Using design patterns in any idea will be considered as a bonus.
- Note that: Bonus items will not be counted unless the original project is complete.

❖ Deliverables for all projects:

- A System with GUI that fulfills the mentioned requirements.
- All projects must be submitted with a printed documentation and a softcopy of your project source code.
- Project documentation should include:
  - The project name.
  - The team members' info (ID, name, and section number).
  - A UML class diagram for the project.

❖ Team members:

- **4 – 6 members per team**.

❖ Teams Submission Forms links for each department:

- Use the links below to submit your team members' information.
- Please make sure that you are in the right department's as submitting in the wrong department will not be counted.
- **Note that, the submission will be closed on Monday 14 of December at 1 pm.**

Teams registration form	https://forms.gle/LjMbhMyptfV7nv189
Students who could not find a team	https://docs.google.com/forms/d/e/1FAIpQLSdPfC3YegfBQQrbUV9oH4HgU2Sc41RkiyHWD1Hb5gEr2WFf3g/viewform

Project ideas registration Forms links for each department:

- After registering your team, you will get your team ID, memorize it.
- Please note that each team will be able to register for only one idea and only once, means that once you submitted your selection you will not be able to update nor submit again.
- Each idea will have maximum number of teams to register in, once an idea is full it will be closed, and no more teams will be able to select it.

Use the link below to register in a project idea	**https://docs.google.com/forms/d/1VE2RNCPzxcTW_BJUxMIdDrhNtfTsVFmcr4I7HOMkXwA/edit?usp=sharing**

- **Note that, the form will be opened on Wednesday 15 of December at 10.00 pm and it will be closed on Friday 18 of December at 11:59 pm.**

Candy Crush Saga

Description

- Candy Crush Saga is a Puzzle Game which is based on Mix and match sweets in a combination of three or more to gain points and other bonuses as you progress. When three or more candies of the same kind are switched next to each other they will burst, and you will gain points.
- The game consists of a grid with size 8*8 so it has 64 spots, where each spot has a candy.
- Basic Entities: Game, Player, Spot**,** Grid, Candy:
- Game Details:
  - The game consists of 1 player.
  - Manages the game (eg game levels).
- Player Details:
  - Starts to mix and match sweets to burst them and gain points.
  - Each player must enter his name and has a score at the end of the game.

- Spot Details:
  - A spot represents one block of the 8×8 grid.
  - Each spot has candy and information about the candies on its adjacent spots (top, down, right, and left).
- Grid Details:
  - The grid consists of several spots.
  - Each spot has a fixed position on the grid.
- Candy Details:
  - We have 6 different regular kinds of candies (Sausage, Lozenge, Teardrop, Pillow, Planet, and Flower), for further details go to this link

Group size

4 - 6 members.

Deliverables

- A System with2D GUI that fulfills the above requirements.
- Project documentation:
  - The project's name.

- The team members' info (ID, name, and section number).

❖ A UML class diagram for the project.

Mentor

❖ TA. "Abdullah Mahmoud"

❖ E-Mail: abdullah_mahmoud@cis.asu.edu.eg

Bonus

❖ Add Special candies with their effect.

❖ Multiple level of difficulties.

❖ Save usernames with their best scores.

Hints

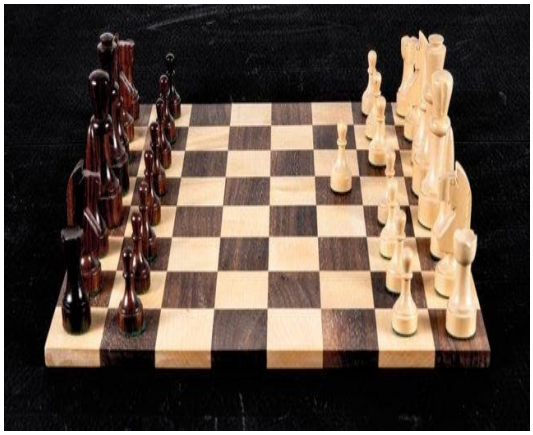
❖ Your project must implement all OOP concepts:

- Encapsulation
- Overloading
- Inheritance
- Polymorphism

Chess Game

Description

- The rules are the same as the standard chess game:
  - Chess is a game played between two players on opposite sides of a board.
  - The chess board contains 64 squares of alternating colors.
  - Each player has 16 pieces: 1 king, 1 queen, 2 rooks, 2 bishops, 2 knights, and 8 pawns.
  - The goal of the game is to checkmate the other king. Checkmate happens when the king is in a position to be captured (in check) and cannot escape from capture.
- Basic Entities: Game, Player, Spot**,** Board, Pieces:
  - Game Details:
    - The game consists of 2 players.
    - Manages the game (eg which player is going to play now).
  - Player Details:
    - Has white or black color (alternative with the other player)
    - Start move pieces to checkmate the other king.
    - The player with white color starts the play.



  - Spot Details:
    - A spot represents one block of the 8×8 grid.
    - Each spot may or may not has a piece.
  - Board Details:
    - The board consists of several spots.
    - Each spot has a fixed position on the board.
  - Pieces Details:
    - We have 6 different kinds of pieces (king, queen, rooks, bishops, knights, and pawns)
    - Each piece has different moves.

Note: The piece's movements, check, and checkmate are required operations.

Group size

4 - 6 members.

Deliverables

- A Game with GUI that fulfills the above requirements.
- Your game should be totally developed in Java language.
- Your project must implement all OOP concepts:
  - Encapsulation
  - Overloading
  - Inheritance
  - Polymorphism
- Project documentation:
  - The project's name.
  - The team members' info (ID, name and section number).
  - The class diagram for the project.

Mentor

- TA. "Abdullah Mahmoud"
- E-Mail: abdullah_mahmoud@cis.asu.edu.eg

Bonus

- Handling other chess rules like Pawn Promotion, and casting.
- Make one player vs Computer.

Hints

- Your project must implement all OOP concepts:
  - Encapsulation
  - Overloading
  - Inheritance
  - Polymorphism

Egyptian League Management System

Description

- Egyptian league management system is a windows application with GUI for managing the Egyptian league matches and teams.
- The system admin has the ability to add/edit/delete a match, team, a player in the team, referee, and stadium.
- Basic Entities: League, Admin, Match, Team, Stadium, Player, Referee:
    - League Details:
        - League information may be (starting date, ending date, list of matches, list of teams)
    - Admin Details:
        - Sign up/in as an admin.
        - Add/ edit/ delete a match, team, a player in the team, referee, and stadium.
    - Match Details:
        - Match information may be (time, date, stadium, match teams, match referee).
    - Team Details:
        - Team information may be (name, list of players, list of matches).
    - Stadium Details:
        - Stadium information may be (name, location, seat capacity, list of matches).
    - Player Details:
        - Player information may be (name, height, weight, team, position)
    - Referee Details:
        - Referee information may be (name, height, weight, list of matches)

Group size

- 4 - 6 members.

Deliverables

- A System with GUI that fulfills the above requirements.
- Project documentation:
    - The project's name.

- The team members' info (ID, name, and section number).

❖ A UML class diagram for the project.

Mentor

❖ TA. "Abdullah Mahmoud"

❖ E-Mail: abdullah_mahmoud@cis.asu.edu.eg

Bonus

❖ Adding new entities that belong to the system (eg information about the team head coach).

❖ Saving the inserted information to the system to load it and append new information to it.

Hints

❖ Your project must implement all OOP concepts:

- Encapsulation
- Overloading
- Inheritance
- Polymorphism

Colon Cancer Diagnosis System

Description

- Colon cancer represents the leading cause of fatality among cancers. Early detection is the only solution that allows treatment before the cancer spreads to other parts of the body. The goal of this project is to build a colon cancer diagnosis system that detects whether the inputted colon cell is a malignant or a benign tumor colon cell, based on the registered cells previously diagnosed. Simple probability and distance equations will be considered. **(All needed formulas will be provided to the team)**

Requirements:

- The user must be able to insert an input to the application, and the application has to classify that input.
- The system must implement 2 classification techniques that can be used to predict whether the patient has Colon Cancer or not. **(All needed formulas will be provided to the team)**
  1. Nearest Neighbor:
     - After classifying the inputted sample by the three independently trained classifiers, the voting module combines their outputs to assign the inputted sample to the most frequent output (class). The output determines whether the inputted sample is infected with colon cancer or non-infected.
     - Using the test samples, you will test the three trained classifiers and separately find the performance of each one using the Overall Accuracy (OA) and the Overall Accuracy of voting model. **(All needed formulas will be provided to the team)**

Group size

4 - 6 members.

Deliverables

- An application with GUI that fulfills the above requirements.
- Your game should be totally developed in Java language.
- Your project must implement all OOP concepts:
  - Encapsulation
  - Overloading
  - Inheritance
  - Polymorphism
- Project documentation:
  - The project's name.
  - The team members' info (ID, name and section number).
  - The class diagram for the project.

Mentor

- T.A. Alaa Tarek
- Email: alaa.tarek@cis.asu.edu.eg

Bonus

- Apply Feature Extraction Technique.

[Dataset]

- The dataset consists of 62 sample-expression vectors obtained from Patients. Each individual sample consists of 200 measured gene-expression levels which best distinguish the two classes of patients in the dataset. These samples represent cells of the colon.
- Forty samples are infected with colon cancer (malignant tumor cells) and the rest of the samples (twenty-two samples) are non-infected (normal cells).
- The training dataset has the first 12 non-infected samples and the first 20 infected samples.
- The testing dataset has the rest of the samples (i.e. the last 10 non-infected samples and 20infected samples).
- The dataset is provided as a text format file, called "colon-cancer dataset".
- As shown in Figure 1, each line represents a different labeled sample and consists of 200 values separated by white spaces, in which the first value indicates the class of the sample which is either 1 (denotes infected cell) or -1 (denotes non-infected cell), and the rest of the values indicate the measured gene-expression levels of the sample.

Class	Gene 1	Gene 2	.....	Gene 200

Figure 1

Dataset link:
https://drive.google.com/file/d/16odmlZMMcd_CwvCVU0V77Qa3eaTQVeP9/view?usp=sharing

- **Classification techniques.**
  2. Nearest Neighbor: using Euclidean distance:
     - Calculate distance between each database sample and the input patient and return sample with minimum value.
     - Euclidean Distance Equation

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}.$$

❖ **Overall Accuracy**

Overall Accuracy=((TN+TP)/Total Samples (n))*100

n=165	**Predicted: NO**	**Predicted: YES**	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Music Store Management System

Description:

- The Music Library Management System has to manage various musical items and comes with a number of models and variety. Maintaining all musical records such as creating order, calculating bills, adding new music, edit the music description, and delete any item from the music library.
- Music Store Manager Program Features
- The program can create order, find music, sold items, item in stock, all items, add new Item, remove Item
  - Create order:
    - The user can create an order, choose items, delete items and buy the item and finally show the price of total items. The user can also choose the same item.
  - Find Music:
    - Users can find their song with different categories Name, Category, Type, and Artist.
  - Sold Items
    - It shows how many items and which items were sold.
  - Item in Stock
    - It shows the item that is in the stock that means the quantity of the item is more than zero.
  - All Items
    - Shows all the items.
  - Show item details.
    - show all details of the item.
  - Add New Item
    - only administrator has permission to add new item after login in
  - Edit Item
    - Edit any item content by administrator only
  - Remove item
    - Can delete any item.
  - Logout
    - logouts from the program.

Group size:

4 - 6 members.

Deliverables

- An application with GUI that fulfills the above requirements.

- Your game should be totally developed in Java language.
- Your project must implement all OOP concepts:
  - Encapsulation
  - Overloading
  - Inheritance
  - Polymorphism
- Project documentation:
  - The project's name.
  - The team members' info (ID, name and section number).
  - The class diagram for the project.

Mentor

- T.A. Alaa Tarek
- Email: alaa.tarek@cis.asu.edu.eg

Bonus

- Use a database to save data.

Hints

- Your project must implement all OOP concepts:
  - Encapsulation
  - Overloading
  - Inheritance
  - Polymorphism

Ticket Reservation Management System

Description:

- Ticket reservation system is a windows application with GUI for an agency. The system enables its clients to reserve tickets for any event. Example of events: matches, concerts, Theaters, Festivals, opening events of any place, conferences, and any kind of events, etc.....
- This system will be used by the call center employees who work inside the agency. So, the client will call the call center employee and ask him to book a ticket for a specific event in a specific time. The employee will search the system if the event is existing. If he found an available ticket for the event, he would do it for the client.
- The users of the system are:
- Administrators:
  - Each Administrator can do the following functionalities:
  - Add/ edit/ delete category of event. Example of category: Match, Theater, Festival, …
  - Add / edit / delete event inside a specific category. Example: in the match category, there is a match which will be played on Sunday 20/12/2020 from 8:00 pm to 10:00 pm at Borj Al Arab stadium etc.
  - Each event has a name, place, time, and description for the event, number of available tickets.
- Employees:
  - Each Employee can do the following functionalities:
  - Add a new registered client in the system. Each client has his personal data. Each client will have a unique serial number.
  - Retrieve data about specific clients by searching for him using his name or by using his serial number. The data which will be retrieved are the client's personal info, his previous events that he attended, the upcoming events that he will attend.
  - Unbook any event of the upcoming events of any client.
  - Remove a client from the system.
  - Retrieve all the events under a specific category.
  - Search event by name.
  - Open the event details.
  - Book event for a client if there is an available ticket.

Group size

4 - 6 members.

Mentor:

- T.A. “Aya Saad”

- Email: aya.saad@cis.asu.edu.eg

Deliverables

- A System with GUI that fulfills the above requirements.
- Project documentation:
  - The project's name.
  - The team members' info (ID, name, and section number).
- A UML class diagram for the project.

Bonus

- Amazing GUI.
- Apply design patterns.
- Any additional (non-trivial) features.
- Use database or file organization to save and retrieve the data.

Hints

- Your project must implement all OOP concepts:
  - Encapsulation
  - Overloading
  - Inheritance
  - Polymorphism

Gym Management System

Description

- Gym management system is a windows application with a GUI that enables the gym employees to manage members' and trainers' data. Also, it enables them to manage activities and classes which are held in the gym.
- The users of the system will be:
- Administrators:
  - Administrator can do the following functionalities:
  - Sign in the system.
  - Add/ edit/ delete any trainer. Each trainer has his personal info and classes that he holds. Also, he has many members that he trains.
  - Assign Trainer to classes based on his availability. For example, trainers cannot be assigned to 2 classes at the same time and day.
  - Open/ edit/ delete classes. Each class has one trainer and a limited number of members. Each class has a description. There are different types of classes.
  - Assign a trainer to the member to keep up with the member's progress.
  - View members in a specific class.
  - View members under a specific type of membership.
  - View all members info.
- Employees:
  - Employee can do the following functionalities:
  - Sign in
  - Add/ edit /delete members. There are 3 types of memberships.
    - Pay as you go (PAYG is more of a "user fee" than a membership. It enables consumers to walk into facility, pay a flat fee and participate in a class or session)
    - Open membership (month-by-month)
    - Term membership (a specific period of time, usually six months or a year)
  - Add members in a class.
  - Remove any member from a class.
  - View any member in a specific class.
  - View members under a specific type of membership.
  - View all members info.

Group size

4 - 6 members.

Deliverables

- A System with GUI that fulfills the above requirements.
- Project documentation:
  - The project's name.
  - The team members' info (ID, name and section number).
- A UML class diagram for the project.

Mentor

- T.A. "Aya Saad"
- Email: aya.saad@cis.asu.edu.eg

Bonus

- Amazing GUI.
- Apply design patterns.
- Any additional (non-trivial) features.
- Use database or file organization to save and retrieve the data.

Hints

- Your project must implement all OOP concepts:
  - Encapsulation
  - Overloading
  - Inheritance
  - Polymorphism

Talabat Clone

Description

- We all know Talabat application (otlob previously). It is an online food ordering system which connects restaurants with hungry customers. The customer should be able to order a meal from a restaurant. You are asked to develop an application with similar functionalities.
- The users of the system will be:
  - **Restaurant Owner:**
    - The restaurant owner is the application user who is responsible for managing the data of a restaurant on the application. A restaurant owner has:
      - Username
      - Password
      - restaurant name
    - His functionalities are:
      - Register/ Login to the system.
      - Add a meal to his restaurant. A meal consists of:
        - Meal Name
        - Meal price
        - Meal Description
        - Meal photo [BONUS]
      - Edit/ remove meals.
      - Browse restaurant's orders. An order consists of:
        - The ordered meal.
        - The quantity required.
        - Additional notes.
        - Order Date.
  - **Customer**
    The customer is the application user who uses the application to order food. A customer has:
    - username
    - password
    - mobile number
    - address
  - His functionalities are:
    - Register/ Login to the system.
    - Browse all of the available restaurants.
    - Browse all of the meals of a specific restaurant.
    - Make an order.
      The order should show the total price (price of meal*quantity)

- Browse all of his orders.

Group size

4 - 6 members.

Deliverables

- A System with GUI that fulfills the above requirements.
- Project documentation:
  - The project name.
  - The team members' info (ID, name and section number).
- A UML class diagram for the project.

Mentor

- TA. Ramy Gamal
- Email: ramy.gamal@cis.asu.edu.eg

Bonus

- Having GUI similar to the actual Talabat application.
- Ability to sort and search meals and restaurants.
- Adding extra features, i.e.: The customer can put meals in a cart and order them.(Be creative)
- Using databases or files to save/retrieve the data.

Hints

- Your project must implement all OOP concepts:
  - Encapsulation
  - Overloading
  - Inheritance
  - Polymorphism

Project Management System

Description

- You are asked to build a system that manages project tasks across the team. The project entities are as follows:
  - **Task:** a task consists of:
    - Task name.
    - Task deadline.
    - Task status, it can be **To-Do** or **On-Going** or **Done** (HINT: use Enums).
    - The task can be assigned to **one or more** users.
    - A user can have **zero or more** tasks assigned to him.
  - **Team Member:**
    - Team members has:
      - username.
      - password.
      - list of assigned tasks.
    - Team member should be able to
      - Browse all the tasks
      - Filter tasks to only see tasks assigned to him.
      - Only be able to edit the tasks **assigned to him**. For example he can change task status from On-Going to Done.
  - **Team Leader**
    - The team leader has all of the attributes and functionalities of the team members plus:
      - Can create/ edit/ delete **any tasks**.
      - Can assign/unassign tasks to himself/other team members.

Group size

4 - 6 members.

Deliverables

- A System with GUI that fulfills the above requirements.
- Project documentation:
  - The project name.
  - The team members' info (ID, name and section number).
- A UML class diagram for the project.

Mentor

- TA. "Ramy Gamal"
- Email: ramy.gamal@cis.asu.edu.eg

Bonus

- Create **Interactive** gui (drag and drop..etc).
- Ability to sort and search tasks.
- Adding extra features, i.e. sending alert emails when the deadline date is close. (Be Creative)
- Using databases or files to save/retrieve the data.

Hints

- Your project must implement all OOP concepts:
  - Encapsulation
  - Overloading
  - Inheritance
  - Polymorphism
- Checkout Trello.

Job Finder Application

Description

- The Job Finder application is a desktop program developed to help job seekers easily browse the best jobs, learn about employers, and get advice on applying for those jobs. It should have the necessary resources to compile a list of the best jobs available. It can be useful whether you are a job seeker or a job poster. A job seeker should be able to browse the available jobs, apply for a job and view his applications. A job poster should be able to view applicants and their information
- Project Required Entities (You should add entities as you need) :
  - Job Application
  - Job Vacancy
  - Job Seeker
  - Company
  - Job Poster
- Required Functionalities:
  - A company has a name, number of employees, job posters, job vacancies and reviews.
  - A job poster has a name, email, password, belongs to a company and has job posts.
  - A job poster can add job vacancies.
  - A job seeker can browse job posts, companies, and their reviews.
  - A job seeker can add a review to a company.
  - A job seeker can apply to a job, update his application or delete it.
  - A job application has a job seeker, an application state.
  - A job vacancy has a list of applications.
  - A job seeker can view the applications and accept or reject them.

- The system should change its behavior based on the type of user signed in, Types of users are:
  - Company Admin (Can add job posters and change company info)
  - Job Poster (Can post jobs, view applications and accept/reject them. He cannot view job seeker's other applications to other companies)
  - Job Seeker (Can view job vacancies, company reviews, apply to jobs. He cannot view job poster info).

Group size

4 - 6 members.

Deliverables

- A System with GUI that fulfills the above requirements.
- Project documentation:
  - The project name.
  - The team members' info (ID, name and section number).
- A UML class diagram for the project.

Mentor

- TA. "Yomna Ahmed"
- Email: yomna.ahmed@cis.asu.edu.eg

Bonus

- Use database or Files to save all data and load them
- Create Mobile or Web application

Hints

- Your project must implement all OOP concepts:
  - Encapsulation
  - Overloading
  - Inheritance
  - Polymorphism

Clinic Management System

Description

- The project clinic management is software developed to simplify the communication process between the doctor and the receptionist while dealing with patients.
- The software would be operated by three users Admin, Receptionist and Doctor.
- Admin would be responsible for adding the others users and can view the profile of each (receptionist, doctor, patient) including personal details like (id, name, address, phone, etc...).
- Receptionist would be responsible for adding patients and assigning token numbers to the patient visiting the clinic and save it in the database along with their details. These token numbers along with respective patient details are sent to doctor.
- The doctor can thus view patient details and after checking up the patient, the prescription which is the recommended medicines for the particular patient is fed into the database by the doctor and is sent to receptionist.
- The receptionist can then view the prescription of patient "that is about patient history" and also can generate bill and feed into the database.
- Note: once receptionist adds token to the patient, the token will be added to the list of tokens and also once doctor adds the prescription for the patient the patient token will be removed from the list of tokens.
- **The user of the system will be:**
  - **Admin** has (user name, password), and his functionalities are (add doctor, add receptionist, view doctor, view receptionist, view patient, change passwords, logout)
  - **Receptionist** has (user name, password), and his functionalities are (add patient, create token, view tokens in lines, view patient, view prescription, discharge patient, change passwords, logout)
  - **Doctor** has (user name, password), and his functionalities are (view patient, add prescription, change passwords, logout)
- Features of the System:
  - **Doctor login:** Doctor has an account in the system from where he checks all patient details.
  - **Receptionist login:** Receptionist is allowed to login and perform token assigning and bill generation.
  - **Token generation:** System automatically generates token number for new arrivals.
  - **Patient information:** Patient information is stored in database along with their prescription.
  - **Bill information:** System generates bill for the patient as requested by the receptionist.

- **Prescription Sending:** Doctor sends prescription to receptionist from where receptionist may provide those medicines to the patient.
- **Resources tracking:** Admin can even maintain a track of resources being utilized in the hospitals by the doctors and patients.
- **Patient bill generation:** The system automatically calculates the patient bill by considering the number of days of stay, and bed and resources charges incurred.
- The system also maintains patient's history so that doctor or receptionist can view them anytime. The project is developed on java language and is supported by a database or file organization to store user specific details and retrieve it when required.

Group size

5-6 members.

Deliverables

- A System with GUI that fulfills the above requirements.
- Project documentation:
  - The project name.
  - The team members' info (ID, name and section number).
- A UML class diagram for the project.

Mentor

- TA. "Hagar Ahmed".
- Email: hagar.ahmed@cis.asu.edu.eg

Bonus

- Amazing GUI.
- Apply design pattern.
- Use database or file organization to save and retrieve the data

Hints

- Your project must implement all OOP concepts:
  - Encapsulation
  - Overloading
  - Inheritance
  - Polymorphism

Library Management System

Description

- The library, contains tens of thousands of books and members which must be organized in order to prevent chaos. Write JAVA program that allow performance of the actions needed in order to manage the library in a simple and comfortable way. The actions will include addition/removal of books, addition/removal of members, member and book searches, and much more.
- The system will support two different types of users.
- Here are the types and the actions available for each type:
- Librarians that can:
    - Addition/Removal of a book from the library
    - Addition/Removal of a user from the library
    - Search for books or members
    - Addition/Removal of users from Book- Order list
    - Blocking of users who return books late
    - Rent a book
- Readers that can:
    - Searching of books or users
    - Addition of self to Book- Order list
    - Rent a book

    - For each user (reader/librarian) exists a unique Id and password, this allows an unambiguous recognition of the user, so that the system can prevent from users access to information which they are not allowed to access.
    - Person should have (ID,Password,Type,FirstName,LastName,Address,CellPhone,Email,isBlocked).

Group size

4 - 6 members.

Deliverables

- A System with GUI that fulfills the above requirements.
- Project documentation:
    - The project name.
    - The team members' info (ID, name and section number).
- A UML class diagram for the project.
- Records of registered users (librarians and readers)
- Records of books

- Records of requests
- Records of loaned books

Mentor

- TA. “Hagar Ahmed”.
- Email: hagar.ahmed@cis.asu.edu.eg

Bonus

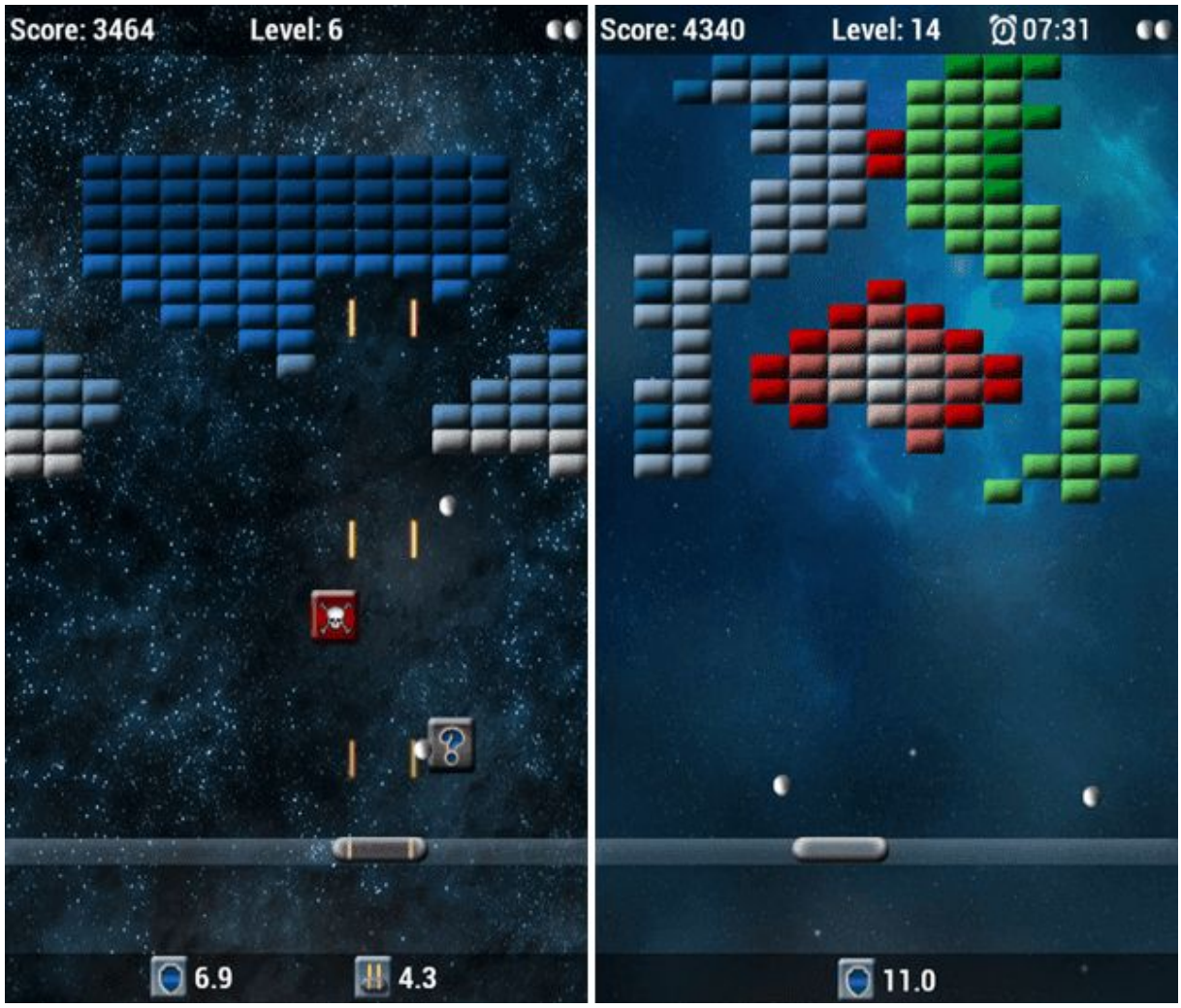
- GUi
- Data is stored in Database

Hints

- Your project must implement all OOP concepts:
  - Encapsulation
  - Overloading
  - Inheritance
  - Polymorphism

Classic Arkanoid Game

Description

- ***Arkanoid*** is an arcade game expanded upon Atari's *Breakout* games (early ball and paddle video games) of the 1970s.
- 

- The Game is based on a random Pattern of Bricks settled together.
- The player controls the "Vaus", a space vessel that acts as the game's "paddle" which prevents a ball from falling from the playing field, attempting to bounce it against a number of bricks.
- The ball striking a brick causes the brick to disappear.
- When all the bricks are gone, the player goes to the next level, where another pattern of bricks appears.
- **Users of the Game:**

- Player: Characterized by Name, Score, High Score.
- **The Game deals with:**
- A Paddle: That is characterized by a Colour, length, speed, Power-ups available,life points.
- A ball: That is characterized by Colour,speed.
- Bricks: Characterized by Colour,collision state,hit counter,Certain bricks, when destroyed, release a power-up —(Gift/surprise) in the form of a falling capsule.
- Power-ups: Characterized by effect code/ID,duration,retrieval state.
- Laser Cannon: a weapon Characterized by ID,laser beam intensity(number of life points reduction),range,duration.
- Shoot-Gun: a weapon characterized by ID, intensity,number of pistols available,duration.
- An enemy: Characterized by speed,life points,weapon ID.
- A board: The container of the bricks,Paddle ,ball,enemy, Player's Score and High Score.
- **Functional Requirements:**
  - Some bricks have to be hit multiple times before destruction.
  - The player's Paddle has to "catch" the Power-up capsule by touching it with the paddle (in this case, termed "Vaus") to retrieve the power-up and acquire its effects.
  - During game-play, pills/capsules fall from destroyed bricks Called Power-ups. When collected, these pills have various effects on the Paddle/Vaus.
  - Among the many Effects the power-ups provided : An increased paddle size, multiple balls, a "sticky" ball (which would stick to the paddle and could be released when the player chose) and even a laser cannon attachment that allows the player's paddle to shoot the bricks.
  - Generate different levels of random bricks distribution in the game board.
  - In the last level you will face an enemy that needs 25 hits to be destroyed.
  - The enemy can have a weapon to hit your paddle and move on the board.

Group size

4 - 6 members.

Deliverables

- A Game with GUI that fulfills the above requirements.
- Project documentation:
  - The project's name.
  - The team members' info (ID, name and section number).
  - UML class diagram for the project.

Mentor

- T.A. Noorhan Khaled
- Email: noorhankhaled1994@gmail.com

Bonus

- Any innovative non-trivial functionality added.
- Any new innovative game Objects or development in the game scenario.
- Use File System or database to aid you store and manage the data in the above functions

Hints

- Your project must implement all OOP concepts:
  - Encapsulation
  - Overloading
  - Inheritance
  - Polymorphism

Examination Management System

Description

- A desktop application that can help instructors to manage and evaluate multiple choice examination questions for students enrolled in different courses. This is done through permitting instructors to add different exam questions for their courses, automating the process of marking and ranking students' responses for these exams, reporting statistics about student grades.

- **The main project user's entities are:**
  - **Instructor:** Characterized by a Username and Password, Name, Mobile number, Email address, Age, list of courses/subject Codes that he teaches.
  - **Student:** Characterized by a Username, Password, Name, ID, Mobile number, Email address, List of taken exams.
  - **Administrator:** Characterized by Username and Password.

- **The main project entities are as follows:**
  - **A Course**
    - That is characterized by an ID, Name, Course_Code (class attribute that can't be changed once assigned), an instructor, a set of added exams.
  - **A Question**
    - That is characterized by a list of options/choices [a,b,c,d], and the Real correct choice stored, a Grade[quantitative mark], An Evaluation rank.
  - **An Exam**
    - That is characterized by an ID, Course_Code, instructor's name, duration, start_time, end_time, Mark, Release_date, list of questions, validation status.
  - **A statistical graph (ex: Histogram)**
    - That is Characterized by a Particular exam Code, list of Grades, the number of students scoring each grade.
  - **An evaluation exam Report**
    - Characterized by Exam Code, the histogram of grades distribution, the top 5 most tough questions experienced in the exam.
  - **Main Functional Requirements**
    - The Admin can login or logout to the system with his username and password.
    - The instructor and the student are required to provide a Username and a Password during the registration process.
    - The instructor and the student can login or log out from the system using their provided username and password.
    - The instructor has the ability to prepare a new exam and insert questions to it.

- The instructor is able to request the system to view an evaluation exam report only after the exam's duration is finished.
- The exam grade is calculated and sent to the student after the duration is finished or the student confirms final submission.
- Questions are ranked automatically after the exam duration ends, according to the most solvable (Experienced >=50% right answers from students) to the Most tough (>50%wrong). This rank can be viewed by the instructor.
- Use File System to aid you store and manage the data in the above functions.

Group size

4 - 6 members.

Deliverables

- A System with GUI that fulfills the above requirements.
- Project documentation:
  - The project name.
  - The team members' info (ID, name and section number).
- A UML class diagram for the project.

Mentor

- TA. Noorhan Khaled
- EMail: Noorhankhaled1994@gmail.com

Bonus

- Powerful interactive Desktop GUI.
- Use database to save all data and load them.
- Make the app operate in an online-basis.
- Adding extra features, i.e, sending an alert message to students when the exam's duration is going to end,
- Create Mobile or Web application.

Hints

- Your project must implement all OOP concepts:
  - Encapsulation
  - Overloading
  - Inheritance
  - Polymorphism

Mini-Payroll System

Description

- A system for managing the data of employees at a company and generating payroll process for each one.
  - An employee should have these attributes: ID, name, age and salary.
  - There are 2 types of employees: "engineer and trainee".
  - An engineer should have these attributes: working hours and a grade.
  - Grade has position, tax rate and pay rate.
- The salary of an employee is calculated using working hours, pay rate and tax rate.
- Pay rate and tax rate are assumed "you could use constant values" according to the position of each employee (Manager, team leader, team member…).
- A trainee should have: a university name, GPA and academic year.
- The salary could be fixed for trainees.
- **System Functionalities:** This application should enable the admin to:
  - Adding new engineer.
  - Edit existing engineer.
  - Delete engineer.
  - Calculate salary.
  - View all engineers with all their data and salaries.
  - Adding new trainee.
  - Edit existing trainee.
  - Delete trainee.
  - View all trainees with all their data and salaries.

Group size

4 - 6 members.

Deliverables

- A System with GUI that fulfills the above requirements.
- Your own system description
- Project documentation:
  - The project name.
  - The team members' info (ID, name and section number).
- A UML class diagram for the project.

Mentor

- TA. "Noorhan Khaled"
- EMail: Noorhankhaled1994@gmail.com

Bonus

- Using files or database to manage the data in all the required functions.
- Extra complicated functionalities.
- Powerful GUI.

Hints

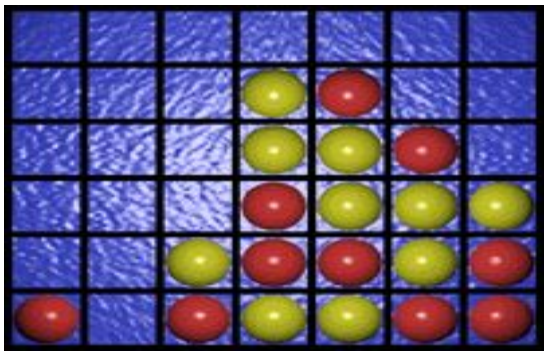
- Your project must implement all OOP concepts:
  - Encapsulation
  - Overloading
  - Inheritance
  - Polymorphism

Connected 4 Game

Description:

- The rules are the same as the standard Connect 4 game:
- Connect Four is a game played between two opponents on opposite sides of a board.
- the players first choose a color.
- The Connect four board contains a seven-column, six-row vertically.
- The objective of the game is to be the first to form a horizontal, vertical, or diagonal line of four of one's own discs.

- Basic Entities: Board, Players, Pieces:
- Board Details:
  - The board consists of 42 empty cells in the first.
  - Use the copy_board function to create a copy of the board before simulating a move.
  - Update cell with the colored piece after play .
  - `find_winner`
- Players Details:
  - The player with Red color starts the play.
  - drop_piece dropping a piece in the board .
  - Dare to Play Again!!!(Y/N)
- Piece Details:
  - Color.
  - Position in the Board.



Group size:

4 - 6 members.

Deliverables

- A System with GUI that fulfils the above requirements.
- Project documentation:
  - 1-The project name.
  - 2- The team members' info (ID, name and section number).
- A UML class diagram for the project.

Mentor

- T.A. Aya Zain
- Email: aya.zain@cis.asu.edu.eg

Bonus

- Make one player vs Computer.
- Attractive Gui.

(Note that this may require you to use the 3D Graphics package)

Hints

- Your project must implement all OOP concepts:
  - Encapsulation
  - Overloading
  - Inheritance
  - Polymorphism

Graph Drawing

Description

As we know that the Graph concept is a very effective concept in Mathematics field and for this importance it becomes a separated field from Mathematics called Graph Theory and this is used widely in solving the Development Problems, Simulating Network Problems , Machine Learning Problems and IOT etc...., so we seek to visualize any graph problem about sending the type of graph, the vertices , the edges and the weights , etc.... to the system. Then the system shall process these received data and shows visualized graph for these data and enable the user to choose some edges from the graph then the system is coloring them and resulting their total sum of weights.

- According; there are various types of graph we may apply that only on Directed Graphs.

Function to deliver

- Create_Graph
- Add_New_Vertex/Vertices
- Delete_Vertex/Vertices
- Delete_Edge / Add Edge
- Update_Weight/Weights
- Show_Graph
- Show-Sum-Of-Some-Weights
- Clear_Graph

Group size

- 4 - 6 members.

Deliverables

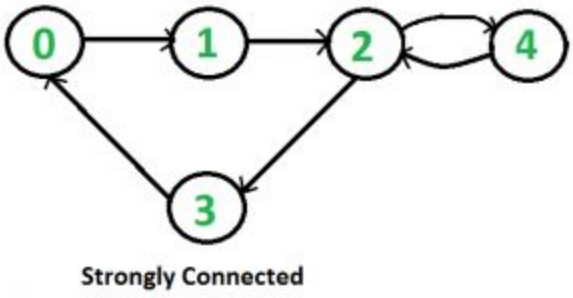
- An application with GUI that fulfills the above requirements.
- Project documentation:
  - The project's name.
  - The team members' info (ID, name and section number).
  - A class diagram for the project.

Mentor

- TA. Aya Zain
- Email: aya.zain@cis.asu.edu.eg

Bonus

- Applying more Types of Graphs.
- Check if a graph is strongly connected



Strongly Connected

Hints

Your project must implement all OOP concepts:

- Encapsulation
- Overloading
- Inheritance
- Polymorphism

Cares Sales System

Description

- Car Sales system is an application that computerizes the conventional car sale procedure which we are aware of. This helps in managing data related to buyers and sellers of the cars.
- This application is divided into following modules:
  - Addition/Deletion of Cars and their details.
  - Viewing and update details
  - Searching for a car.
  - Buying a car.

Addition/Deletion of Cars and their details

- System shall allow the administrator to login and add data related to cars.
- He will also be able to delete old data related to cars.
- Admin has the permission to add, edit and delete data.
- Shall be able to save list of cars.

Viewing details

- View the saved list of cars.
- View all details of a car like year of purchase, manufacturer type, model name, available colors, ...

Searching for a car

- Search for cars based on matched keywords such as cost, year of manufacture etc. and view their details.

Buying a car

- User shall sign up to use the system or login if he/she has an account.
- User can view list of cars.
- User can select one of the cars and buy it.
  - .

Group size

- 4 - 6 members.

Deliverables

- An application with GUI that fulfills the above requirements.
- Project documentation:
  - The project's name.
  - The team members' info (ID, name and section number).
  - A class diagram for the project.

Mentor

- TA. Aya zain
- Email: aya.zain@cis.asu.edu.eg

Bonus

- Add image to each car and display it when you search for that car.

Hints

Your project must implement all OOP concepts:

- Encapsulation
- Overloading
- Inheritance
- Polymorphism

Football Mini Fantasy

Description

A game of prediction, each user selects a team of 11 players at the beginning and gets his points based on each player's performance by the end of league. Each position of a player has his own formula to calculate that player points. We will have -at the end- top 3 users that have the most total points and the top player that had the most points of every user sorted from the highest to the lowest, also can filter the players that had at least X points.

The game has:

1- Players of different positions, (Goalkeepers, Defenders, Midfielders, Strikers):
   a. Each player has his name, his club name and league performance attributes such as how many goals he scored, assists he made and clean-sheets he had.
   b. Can calculate his points based on his performance:
      i. Striker: (goal = 3 points, assist = 4 points, clean-sheet = 1 point).

ii. Midfielder: (goal = 4 points, assist = 3 points, clean-sheet = 2 point).
iii. Defender: (goal = 7 points, assist = 5 points, clean-sheet = 4 point).
iv. Goalkeeper: (goal = 10 points, assist = 7 points, clean-sheet = 3 point).

2- Users:
   a. Each user has a name, password and an array or a list of 11 players that the user can select.
   b. Can create a team by selecting 11 players from a list of all the available players
   c. Can calculate the team's total points.

3- League:
   a. Multiple users that can be added to.
   b. Can get the top 3 users that have the most points sorted from the highest.
   c. Can get the top player for every user and sort them.
   d. Can get each player that had the at least X points, displays them with the team name and the user that had selected them.

Notes:

1. The player is unique for the team, in other words, no two users can select the same player.
2. The selecting player list can only display the player name and club name but not the performance attributes since the game is based on the predictions.
3. The user can select at most 3 players that play in the same club.
4. Take care of your access modifiers of course and don't let any object access other objects data without controlling this access.

Player data example:
-> name: Salah, culbName: Liverpool, goals: 32, assists: 12, clean-sheets: 6.

Group size

4- 6 members.

Deliverables

- A System with GUI that fulfills the above requirements.
- Project documentation:
  - The project name.
  - The team members' info (ID, name and section number).
- A UML class diagram for the project.

Mentor

- TA. Abdelrahman Mosa'ed
- Email: abdelrahman.mosa3ed@cis.asu.edu.eg

[Bonus]

- Shiny GUI that seems like a real game.
- Any used design pattern.
- Read&Write from a Database.
- Any new feature that doesn't violate any of the OOP concepts.

[Hints]

- Your project must implement all OOP concepts:
  - Encapsulation
  - Overloading
  - Inheritance
  - Polymorphism
- Players data can be entered through a simple form or from a database as a bonus.
- There is a form to create a user and let him/her select his/her team, and that form can be accessed after filling the players data.
- There should be a form to let you select what the game does from the functionality that each object has such as displaying top 3 users, or the top 3 players sorted or filtering the players ... etc.

Daily-Moments Book

Description

A GUI desktop application to save your daily moments in which you can go back to and do some filtration on them so you can remember your happy or sad memories. User will choose to sign up as a user or a premium user, then he can login and use the application freely.

The application has:

1. Moment:
    a. has a name, year that it has been in, description and importance rate from 1 to 5.
2. Memory which is a Moment:
    a. also has a location that has been happened at, happiness points, sadness points.
    b. can calculate how happy it was by:
        i. (happiness points - sadness points) * importance rate.
3. Achievement which is also a Moment:
    a. also has proud points, the year that it was planned to happen in.
    b. can calculate how happy it was by:
        i. (proud points + 10 * max(1, the year that it was planned to happen in - the year that it has been in)) * importance rate.
4. User:
    a. has a username,password, nickname and a timeline.
    b. can login to the system at the beginning
    c. can logout from the system (bonus save his data into a database so when he logs in again, you can get his saved moments)
    d. can add/edit/delete a new moment to/from his timeline.
    e. can filter the moments that are above or below X importance rate.
    f. can sort the moments asc/desc by importance rate and can filter those moments based on the year that they'd happened at.
5. PremiumUser is a User that can do all the user functionalities and also:
    a. can get the average importance rate for all of his/her previous moments.
    b. can get the average importance rate for all of his/her previous moments in X year.
    c. can get his/her top rated moment.
    d. can get his/her least rated moment.
    e. can get his/her happiest moment.
    f. can get his/her saddest moment. (least happy moment)

Group size

4- 6 members.

Deliverables

- A System with GUI that fulfills the above requirements.

- Project documentation:
  - The project name.
  - The team members' info (ID, name and section number).
- A UML class diagram for the project.

Mentor

- TA. Abdelrahman Mosa'ed
- Email: abdelrahman.mosa3ed@cis.asu.edu.eg

[Bonus]

- Shiny GUI.
- Any used design pattern.
- Read&Write from a Database.
- Any new feature that doesn't violate any of the OOP concepts.

[Hints]

- Your project must implement all OOP concepts:
  - Encapsulation
  - Overloading
  - Inheritance
  - Polymorphism
- At the beginning, when you create a user, you choose if he will be a premium user or not. **(also the premium user is a class not an attribute inside the User)**
- Kindly try to separate your function so that each function does its job, not multiple jobs.
- It would be great if each GUI form does a simple job too.