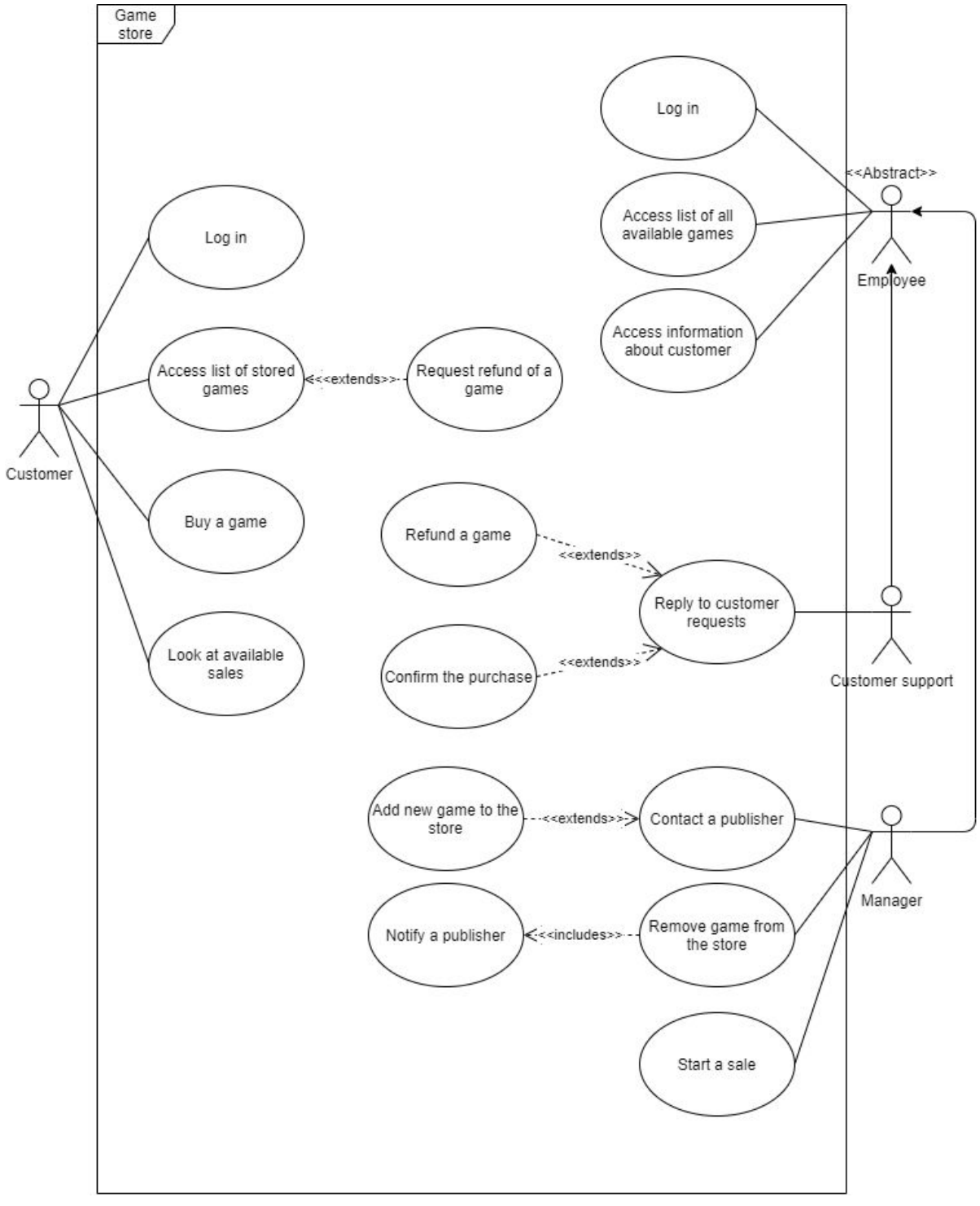
Application for selling and managing games

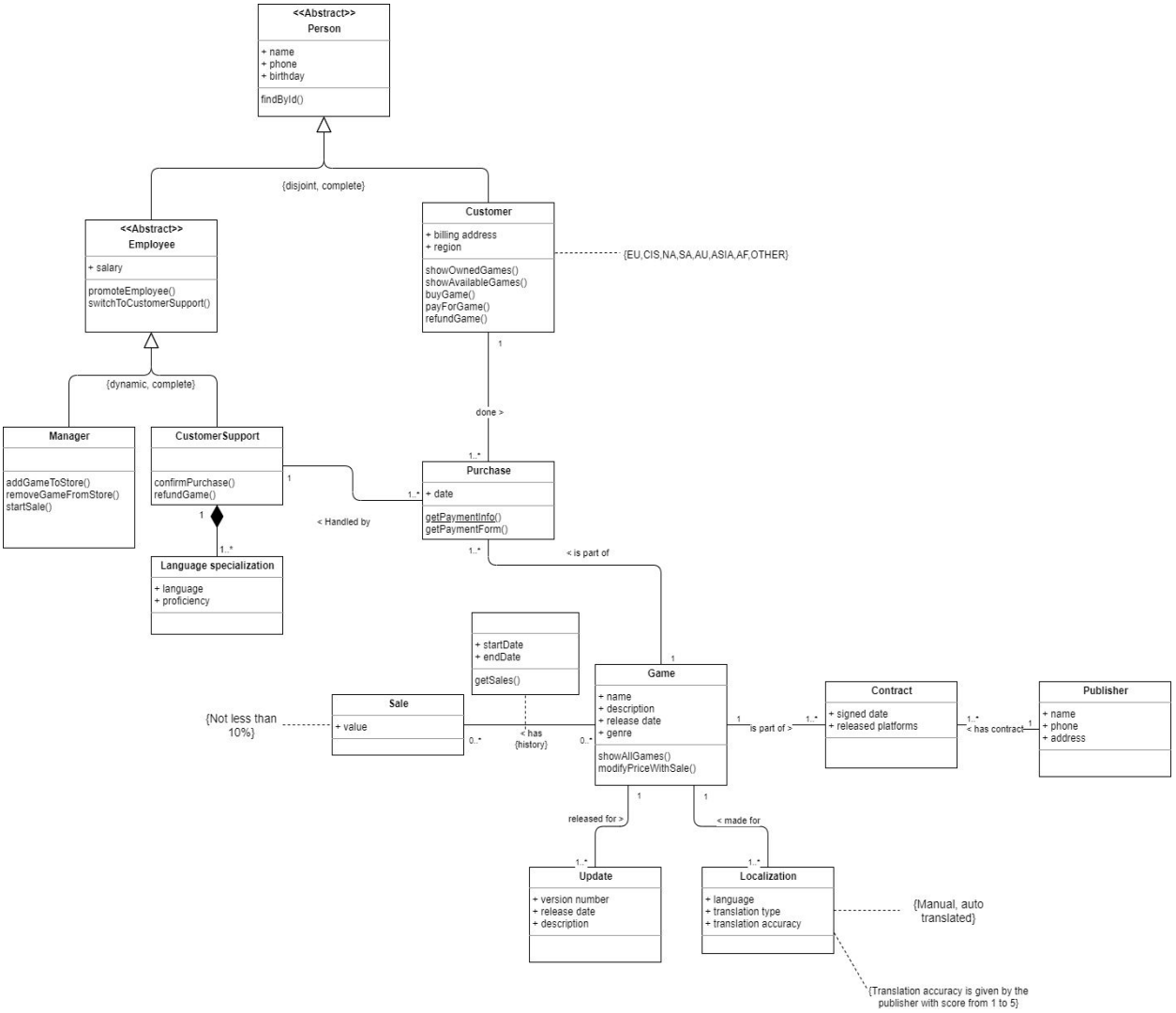
User requirements

1. Application that handles the management and sales of games.
2. Application can be used by customers and employees.
3. A customer can order to buy a game by specifying their personal information.
4. An order from a customer gets accepted by customer support and handles the purchase approval.
5. A manager can add/remove games from the store and organize sales for certain games on a given date.

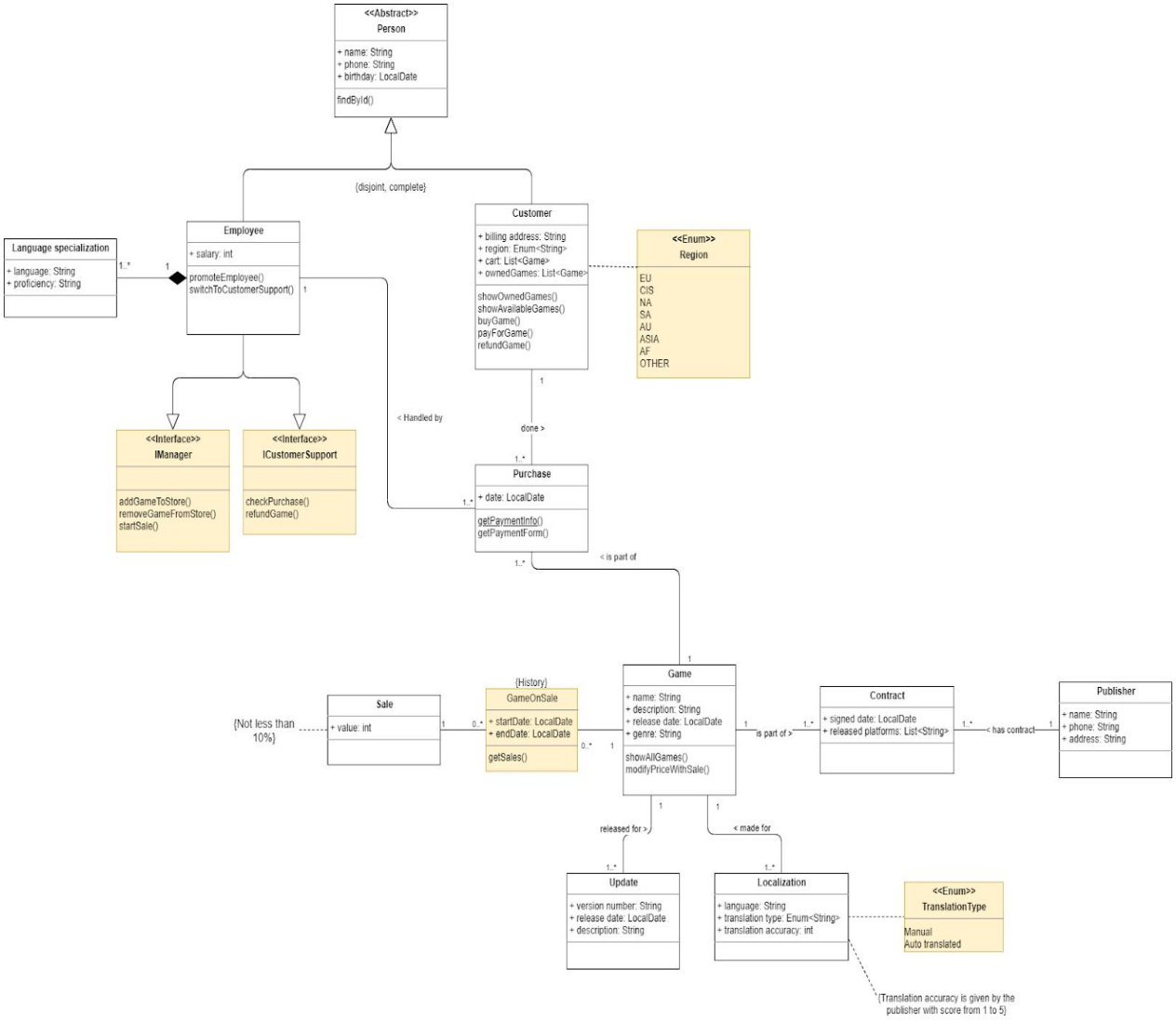
Use case diagram



Analytical Class Diagram



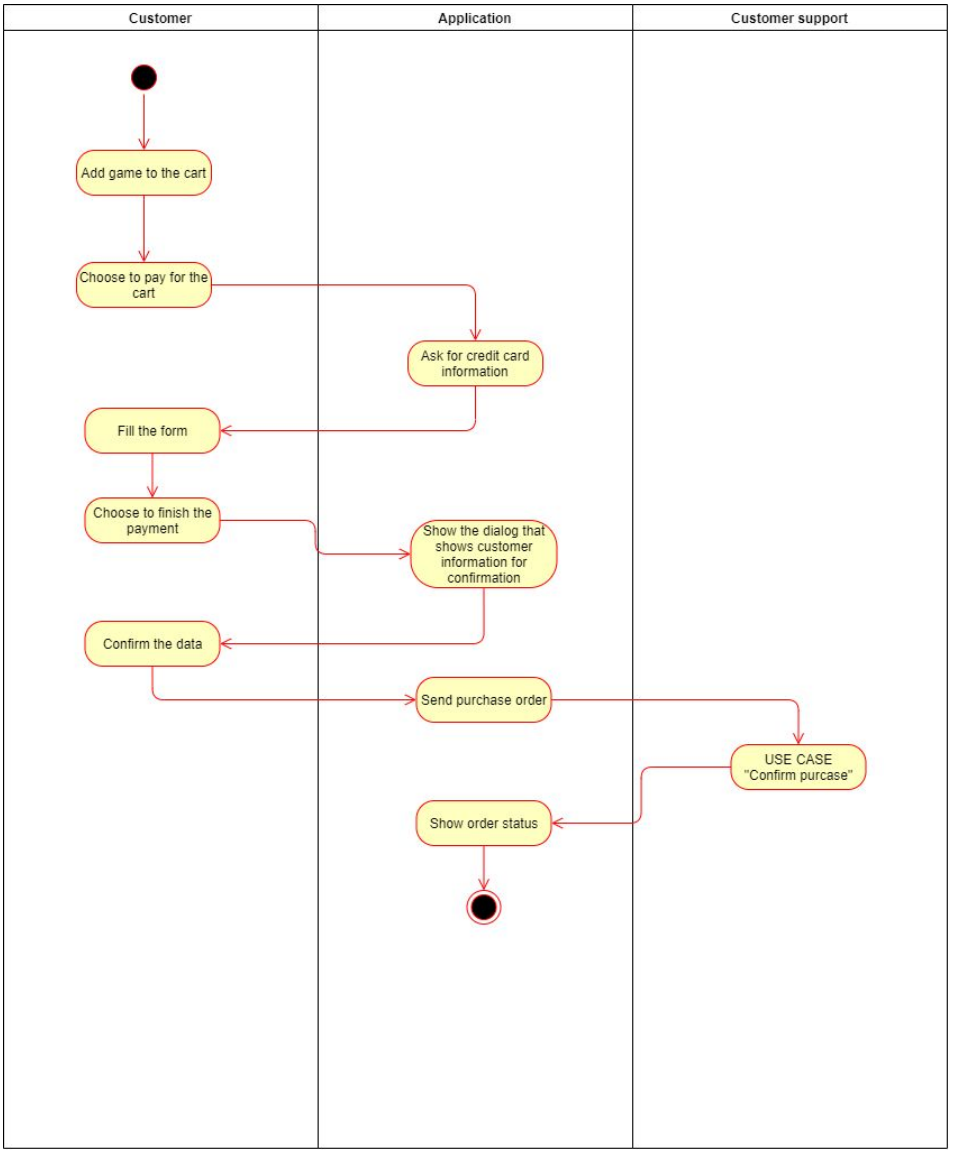
Design Class Diagram



The scenario of selected use case

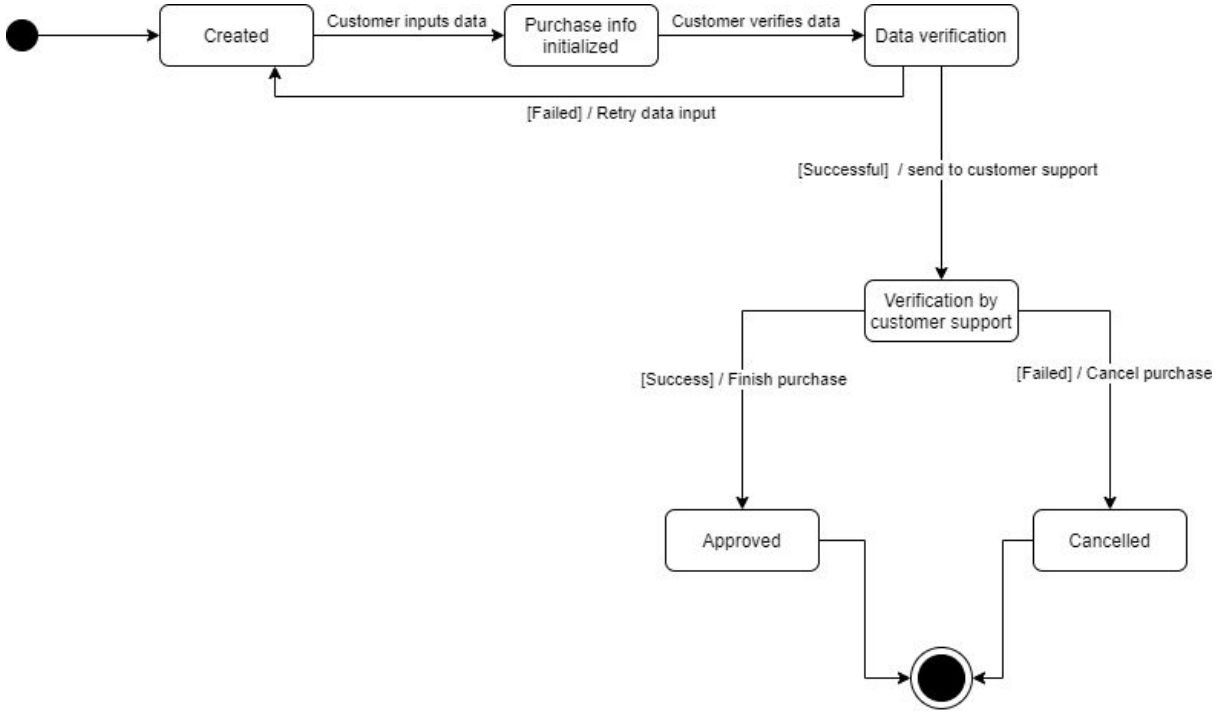
Customer buys a game.
Customer first selects the game they want to buy. That game is added to the cart. As the customer proceeds to pay for the items in the cart, they are asked to specify the information about the credit card and the billing address. As the customer is ready to send the order, they are first asked to double check the given information. After the customer has confirmed that the data is correct, an order is sent to the customer support. App will show the order status based on the response of the customer support.

The activity diagram

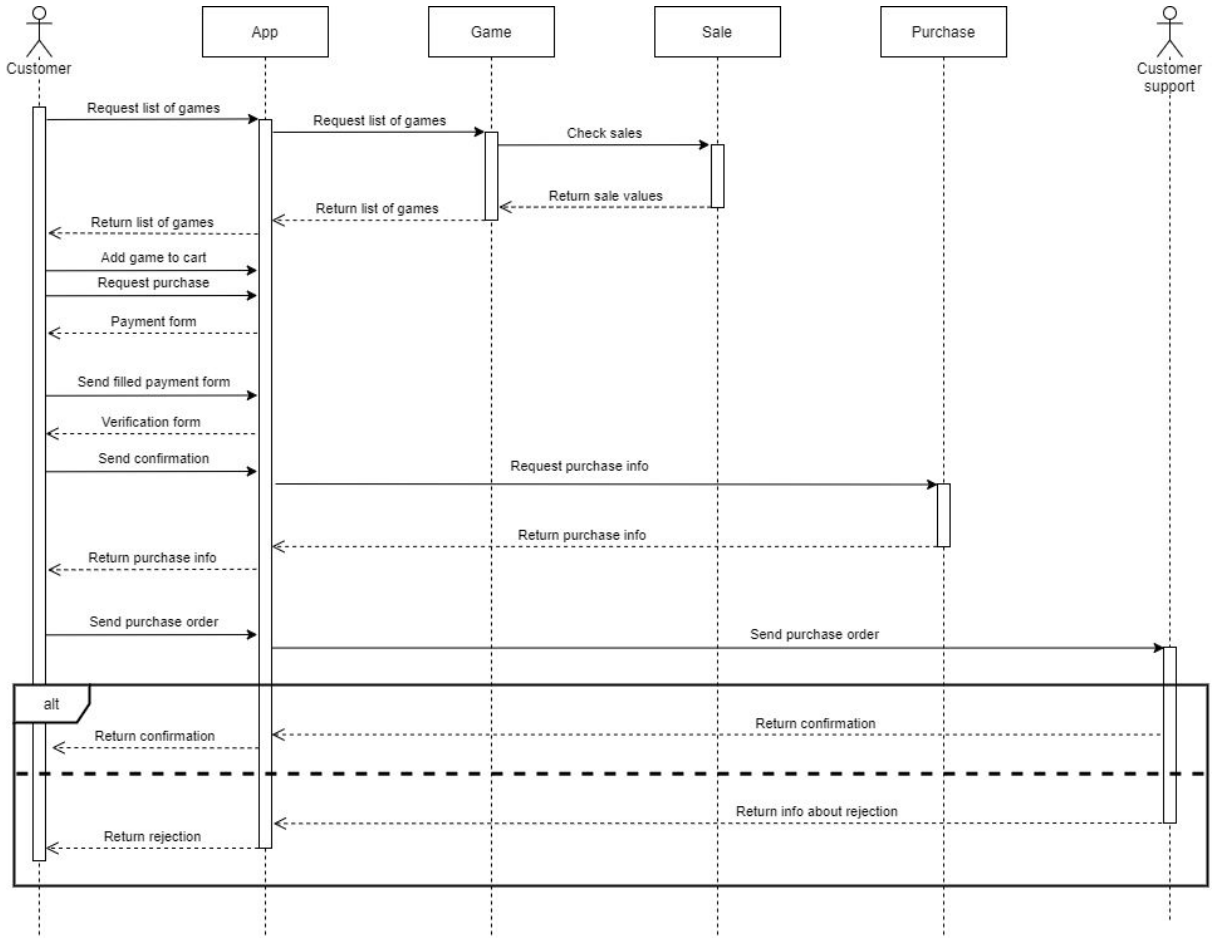


The state diagram

Purchase class

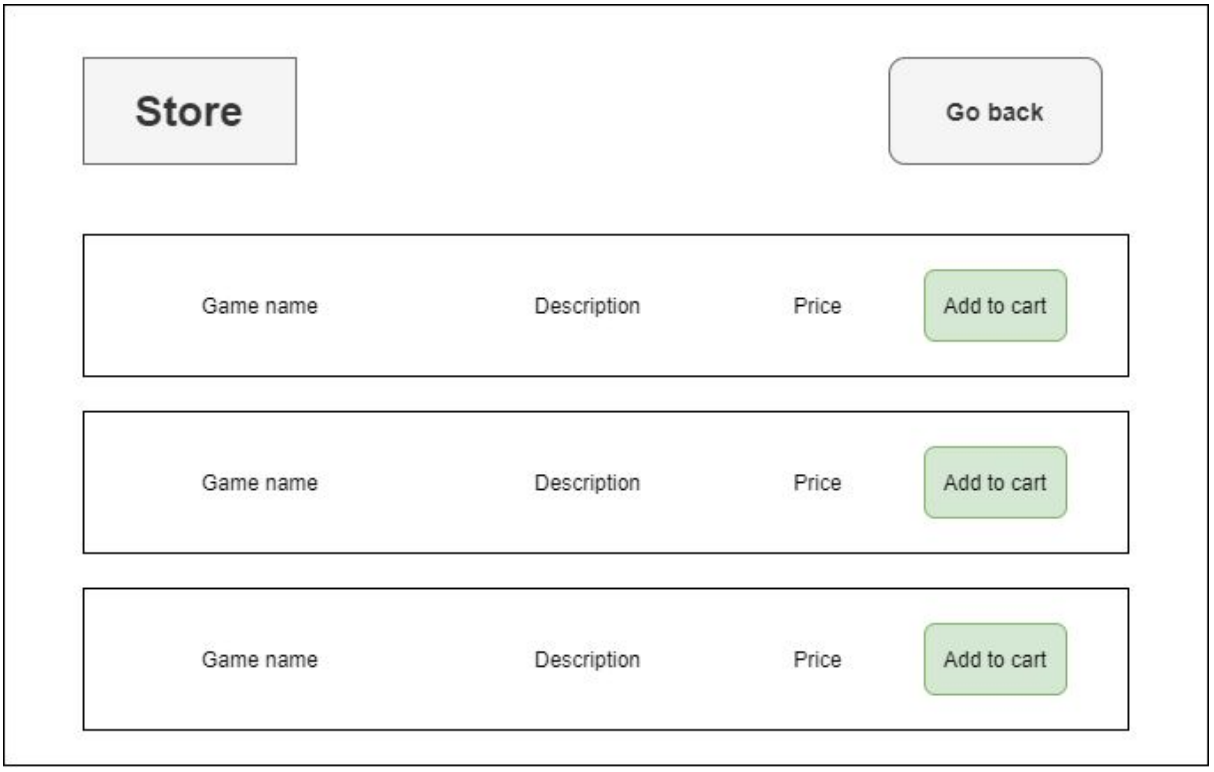


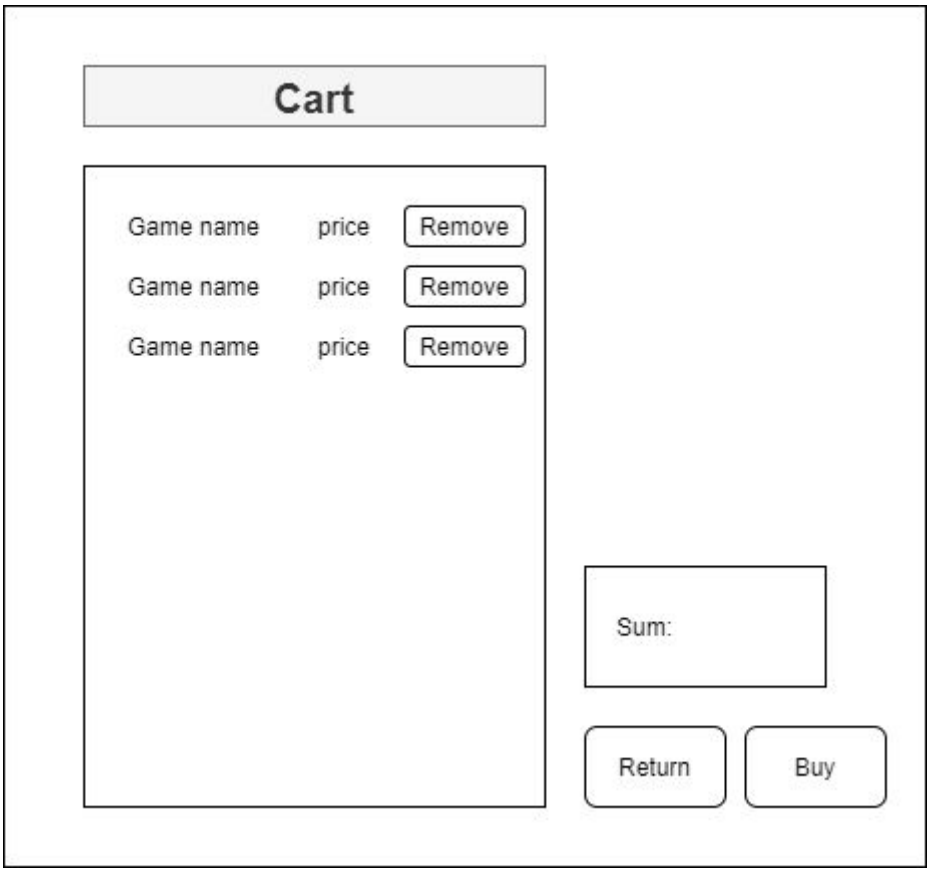
The interaction/sequence diagram



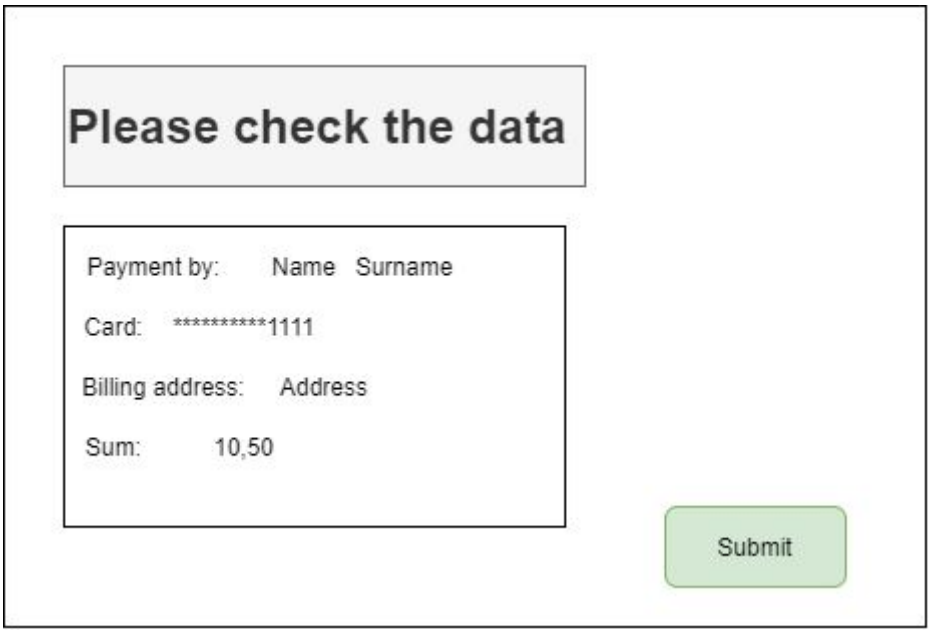
GUI


Owned games
Go to store
Game name
Description
Download
Game name
Description
Download
Game name
Description
Download


Store
Go back
Game name
Description
Price
Add to cart
Game name
Description
Price
Add to cart
Game name
Description
Price
Add to cart


Cart
Game name
price
Remove
Game name
price
Remove
Game name
price
Remove
Sum:
Return
Buy


Payment
Credit number
Expiration date
MM
YY
Name
Surname
Billing address
Submit



The discussion of design decisions and the effect of dynamic analysis

Dynamic inheritance of Employee class

Dynamic inheritance means that our children classes can switch between each other. In our case, a manager can take over the job as customer support(for various reasons like no customer support is available at that time) meanwhile customer support can be promoted to manager. Dynamic inheritance is implemented by utilizing interfaces for child classes. Our employee class will have all attributes needed for both classes(including connection to language class).

History association(class GameOnSale) between Sale and Game

History association allows us to reuse the same object in association with another class multiple times using date as a context. In our case, GameOnSale class allows us to define the date when a sale(with certain value) starts and ends for a given game. History association is implemented by having an association class GameOnSale

Composition of Language class

Instances of language classes can not exist without their parent class. In our case Language class is part of Employee class. Originally in analysis diagram, language is connected to Customer Support but as we utilize interfaces for inheritance, the connection is moved to Employee class. The composition can only be made if the Employee object is defined with Customer support Interface. Composition is implemented by having 2 extends that make sure that a Language object belongs to only one Employee and will be deleted if an Employee gets removed. Constructor is private.