Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

Predmet / Subject

– Modelovanie softvéru / Software modeling –

- Dokumentácia / Documentation -

Collector Coin Exchange

Ak. Rok / Academic term: 2021/2022, zimný semester

Cvičiaci / Instructors:
doc. Ing. Valentino Vranić, PhD.
Mgr. Pavle Dakić

Študent / Student:
Lukáš Peťko

Bratislava, 2021.

Obsah / Content:

1 Zámer projektu / Project intent

Collector Coin Exchange

Numismatics, coin collecting, exists since the beginning of the 19th century. Even though it is quite common and popular, it is sometimes difficult to get to some coins, mainly the rare ones. That's why I think that users will find the system where they can freely trade and bid their coins useful.

Users can either put one of their coins/coin sets into the auction, or they can bid to another existing auction. Auctions can be planned on an exact date and time to attract a wider audience. The system will track the most viewed auctions and display them to Users as "auctions of the day".

The application will use the traditional English type of auction, where can seller choose if he wants his auction to be nonmoderated (the auction will have set its end which will get prolonged every time someone bids, so everyone has time to react to the new price), or the user can also hire one of the professional moderators for his auction.

Users can also create a wishlist, where they can save the coins they desire and if one of them gets auctioned, they will get notified, and also users can set the ceiling for automated bidding so they don't have to respond immediately.

2 Prípady použitia / Use cases

2.1 UC Create new auction

- Creation of the new auction by the auctioneer

1. The Seller wants to create a new auction
2. If the Seller is logged in the System fills all the user information from the profile
3. If the Seller is not logged in the Seller is asked to log in
   UC Login User
4. The System asks the Seller to fill in information about the auction
5. The Seller fills name of the coin/coin set
6. The Seller fills the category of the coin (Antique, Collector,…)
7. The Seller fills condition of the coin
8. The Seller fills in moderation type and starts the auction
9. The System confirms that all needed information was filled
10. UC ends

2.2 UC Update current auctions

- The use case that covers the situation where the user wants to update information in one of his auctions

1. The Seller wants to update one of his auctions
2. The Seller selects the auction he wants to update
3. The System lists all information about the auction
4. The Seller edits the fields he wants to update
5. The Seller confirms the changes
6. The System saves the changes

2.3 UC Search for an auction by name

- The use case that covers Buyer searching for an auction

1. The Buyer wants to search for an auction
2. The System shows search area
3. The Buyer inserts name of the coin he wants to find
4. The System lists auctions based search results

2.4 UC Filter auctions

- The use case that covers Buyer filtering auctions

1. The Buyer wants to filter auctions
2. The System shows filter options (age of the coin, price, condition,…)
3. The Buyer picks filters
4. The System filters auctions

2.5 UC Bid to an existing auction

- The use case that covers the User's bidding to an auction

1. The Buyer wants to bid to an auction
2. The System lists all available auctions
3. The Buyer selects desired auction
4. The System shows auction information and current price
5. The Buyer can either bid the lowest possible increment or choose the new price (higher than the lowest possible increment)
6. The System saves the new price
7. UC ends

2.6 UC Contact auction winner

- The use case that covers actions at the end of the auction

1. The System contacts the highest bidder (Buyer)
2. If the Buyer responds within 1 day he wins the auction
   UC Win the auction
3. If the Buyer does not respond within 1 day, the System contacts the next highest bidder, and UC goes back to point 2
4. UC ends

2.7 UC Win the auction

- The use case that covers coin payment and delivery

1. The System prompts the Buyer to pay for the coin/coins
2. The Buyer pays
3. After receiving the payment, the System asks the Buyer the delivery options
4. The Buyer fills in the delivery options
5. The Systems sends a request to ship the coin/coins to the User
6. UC ends

2.8 UC Login User

1. The User wants to log in
2. The System asks the user to log in to an existing account or to create a new account

3. If the user already has an account, he fills in username and password
4. If the User does not have an account he creates one
   UC Create account
5. The System verifies username and password and logs in to the User
6. UC ends

2.9 UC Create Account

1. The User wants to create an account
2. The System asks the User to fill in his personal information (name, address, username, password)
3. The User fills in his personal information
4. The System verifies filled information
5. UC ends

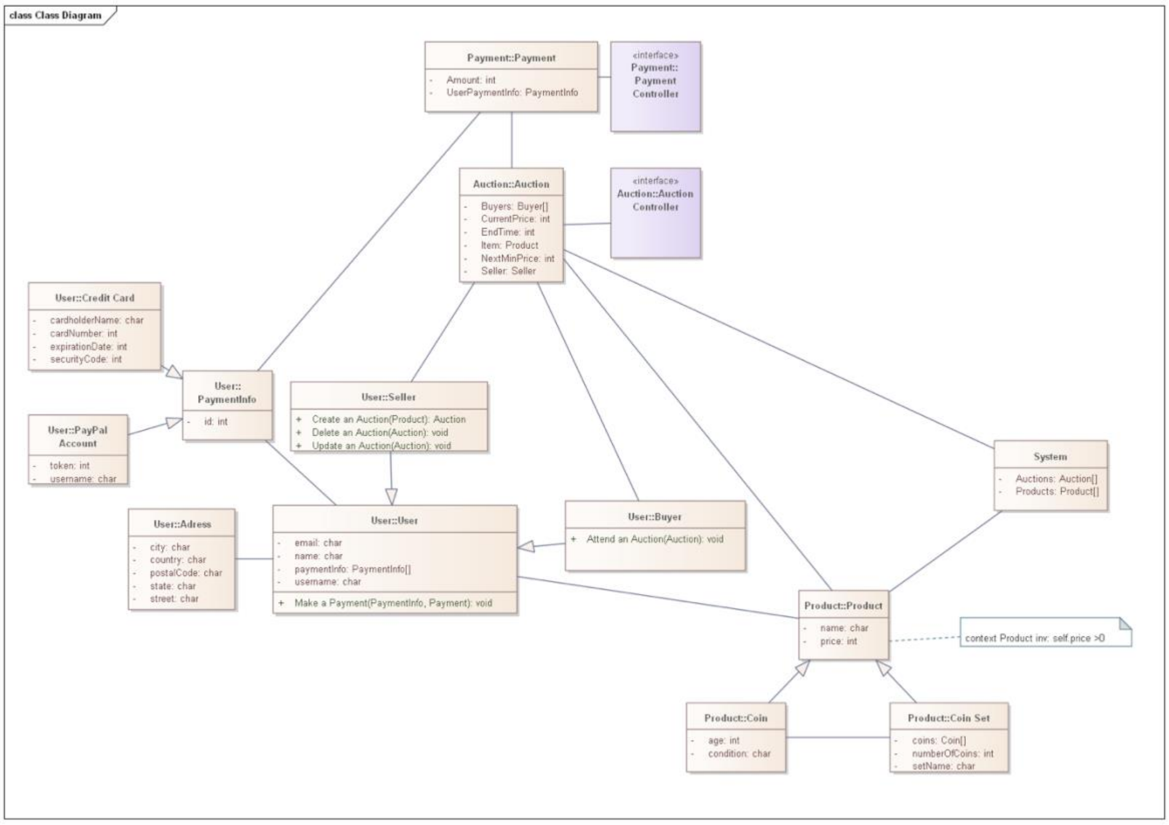
3 Iniciálny model správania v UML / Initial behavior model in UML

In this part, Use case model for all the usecases mentioned above. I also created activity, sequential and Collaboration diagrams for the following Use Cases: UC create new auction, UC Bid to an existing auction, UC contact the winner.

4 Objekty, triedy a interakcia / Objects, classes and interactions

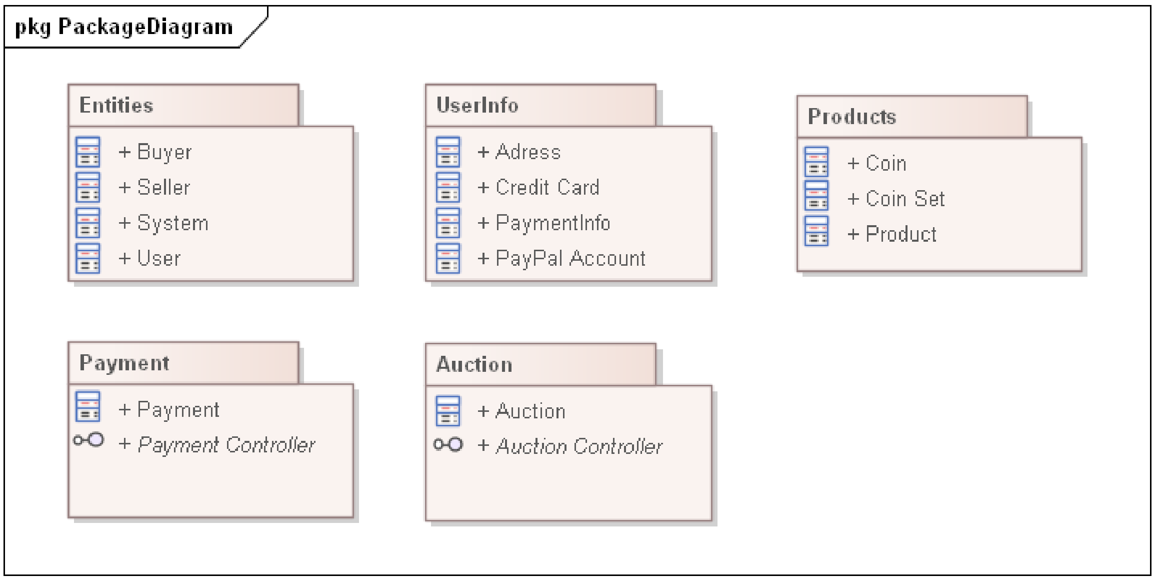
4.1 Class Diagram + OCL

In this section I created class diagram of all classes, that will be needed for our project to run. I also added an OCL to the price of the Product class.



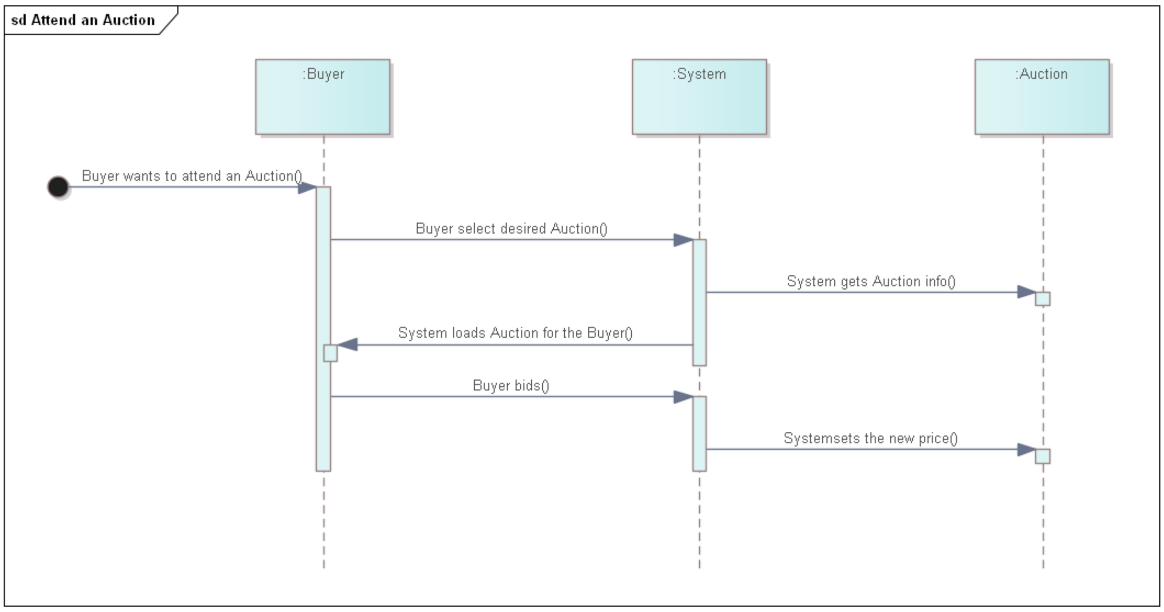
4.2 Package diagram

This diagram shows, how multiple classes are grouped together

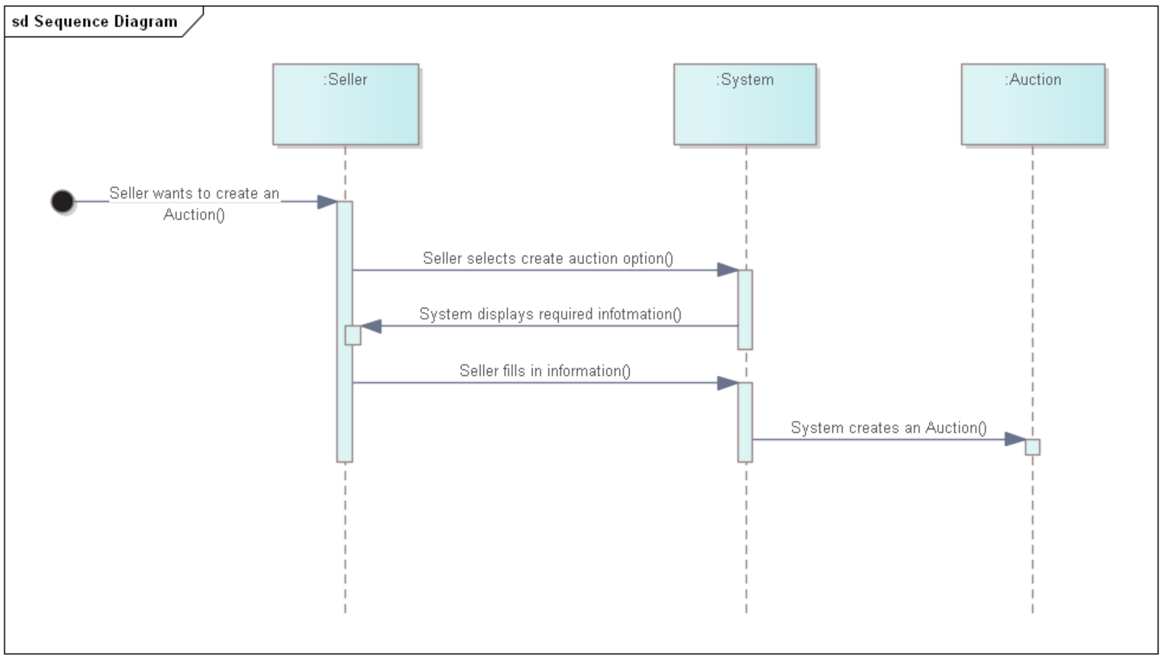


4.3 Method Sequence Diagram

These diagrams show flow of data between objects.

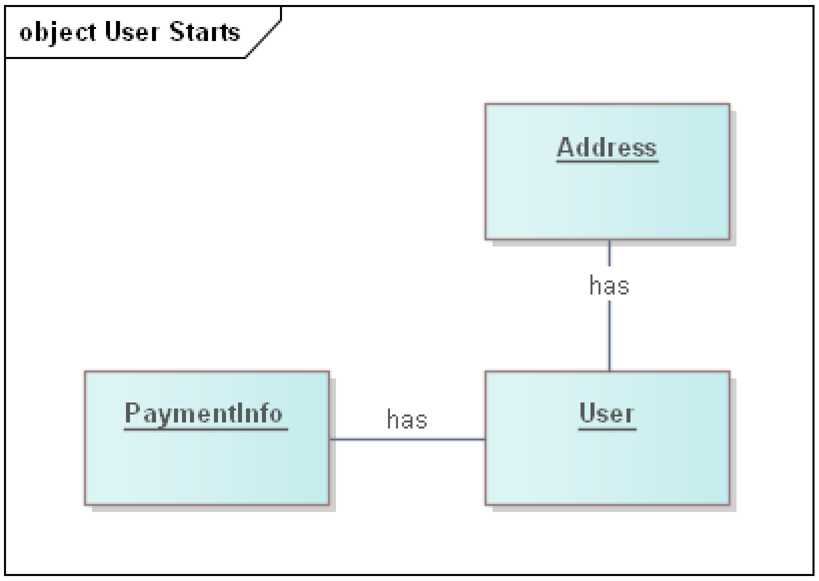

sd Attend an Auction
:Buyer
:System
:Auction
Buyer wants to attend an Auction()
Buyer select desired Auction()
System gets Auction info()
System loads Auction for the Buyer()
Buyer bids()
Systemsets the new price()

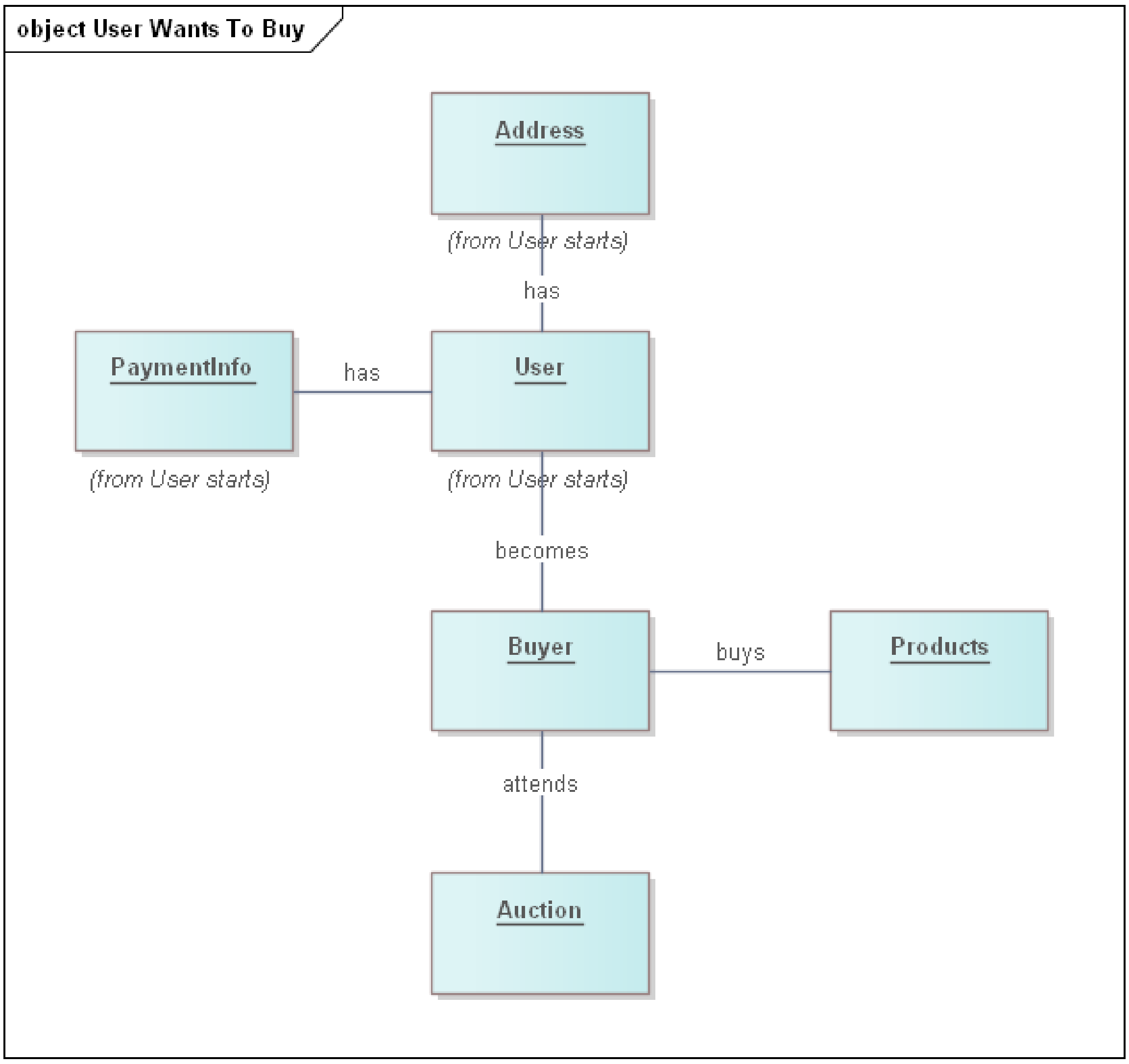
Create an Auction

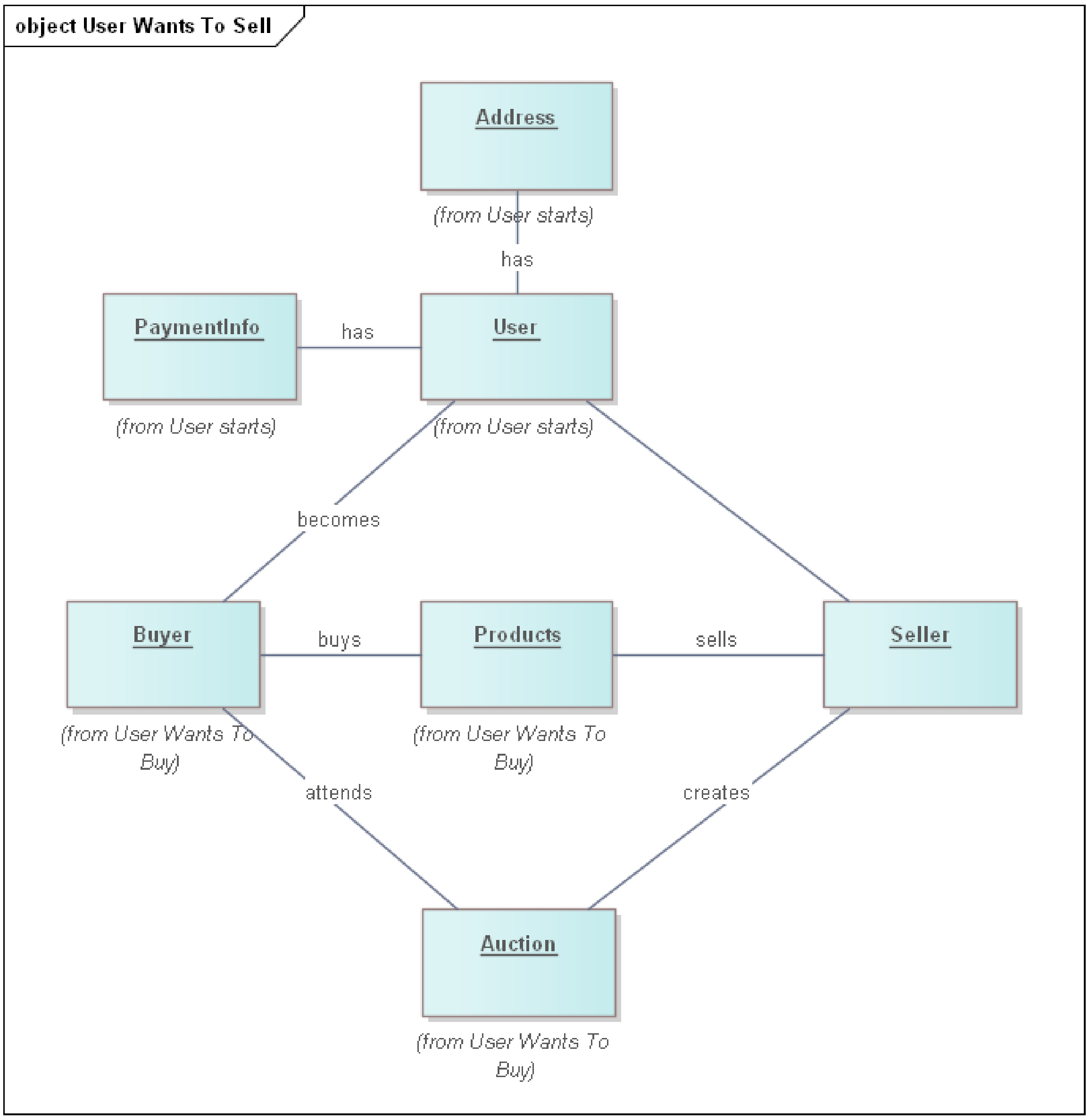

sd Sequence Diagram
:Seller
:System
:Auction
Seller wants to create an Auction()
Seller selects create auction option()
System displays required infotmation()
Seller fills in information()
System creates an Auction()

4.4 Object diagram

These diagrams show which objects are created during various processes.

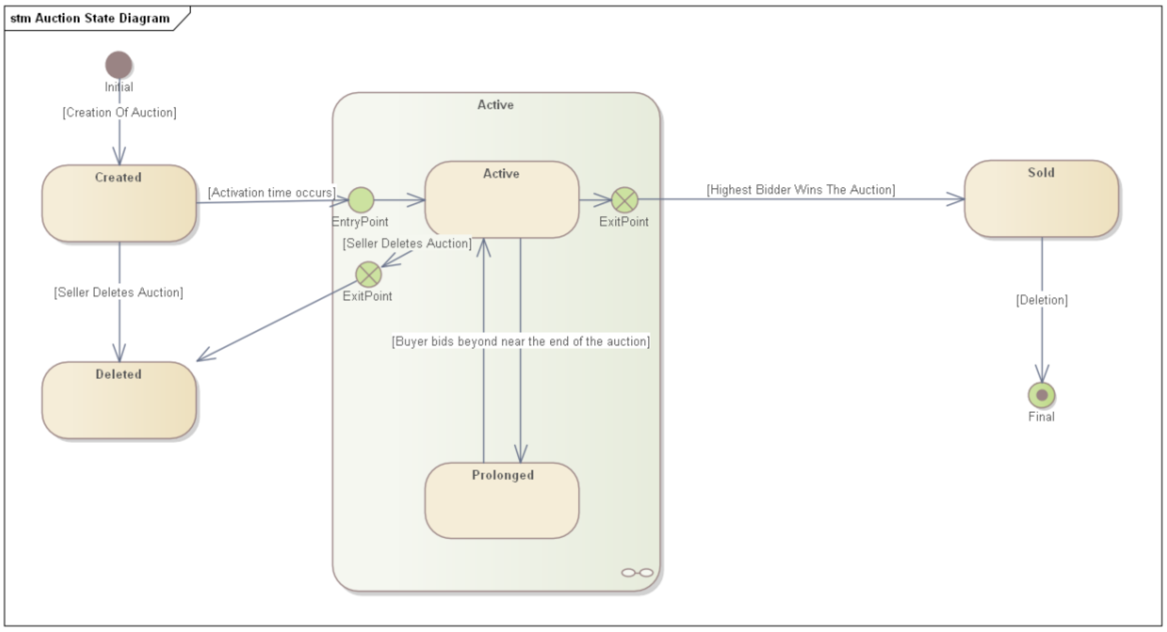


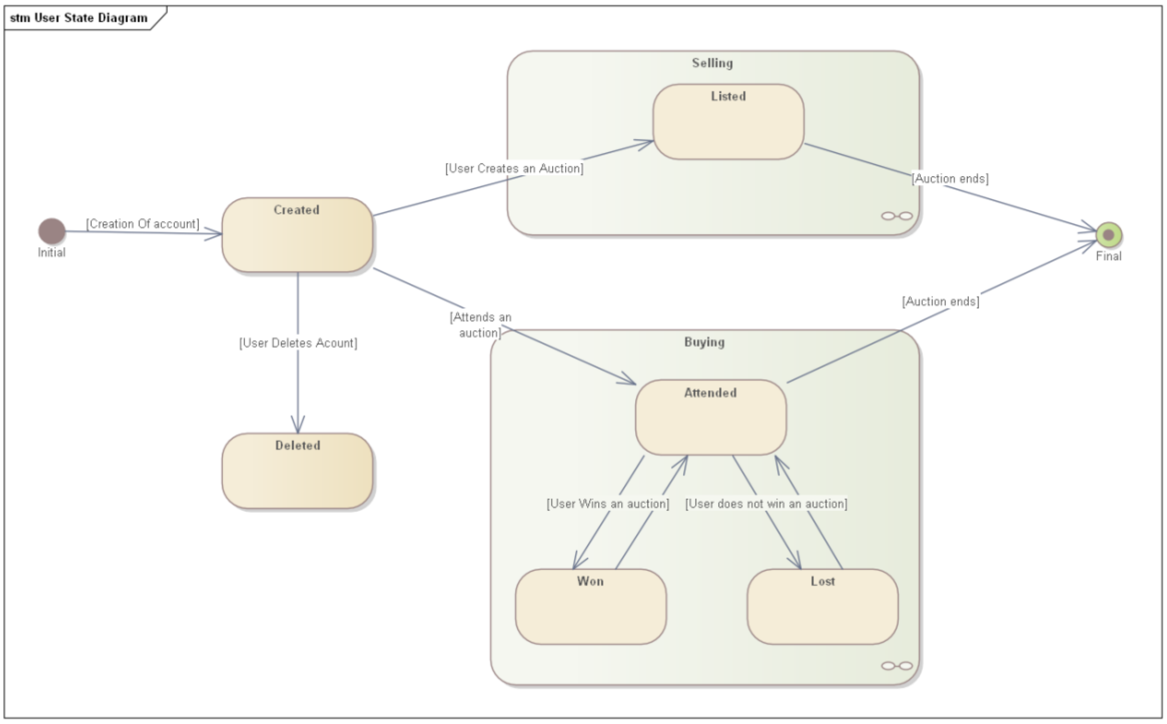

object User Wants To Buy
Address
(from User starts)
has
PaymentInfo
has
User
(from User starts)
(from User starts)
becomes
Buyer
buys
Products
attends
Auction



4.5 State diagram

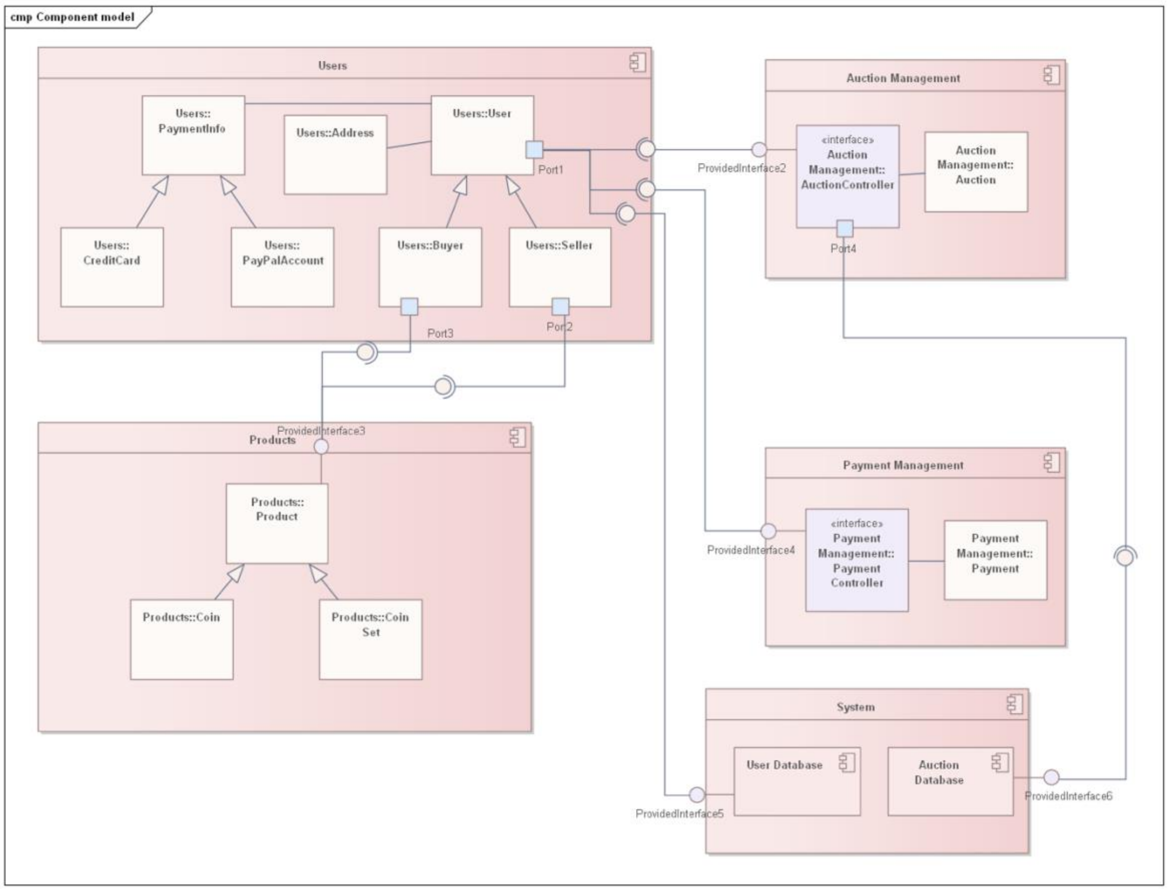
These diagrams show what states can entities have during processes of the program.



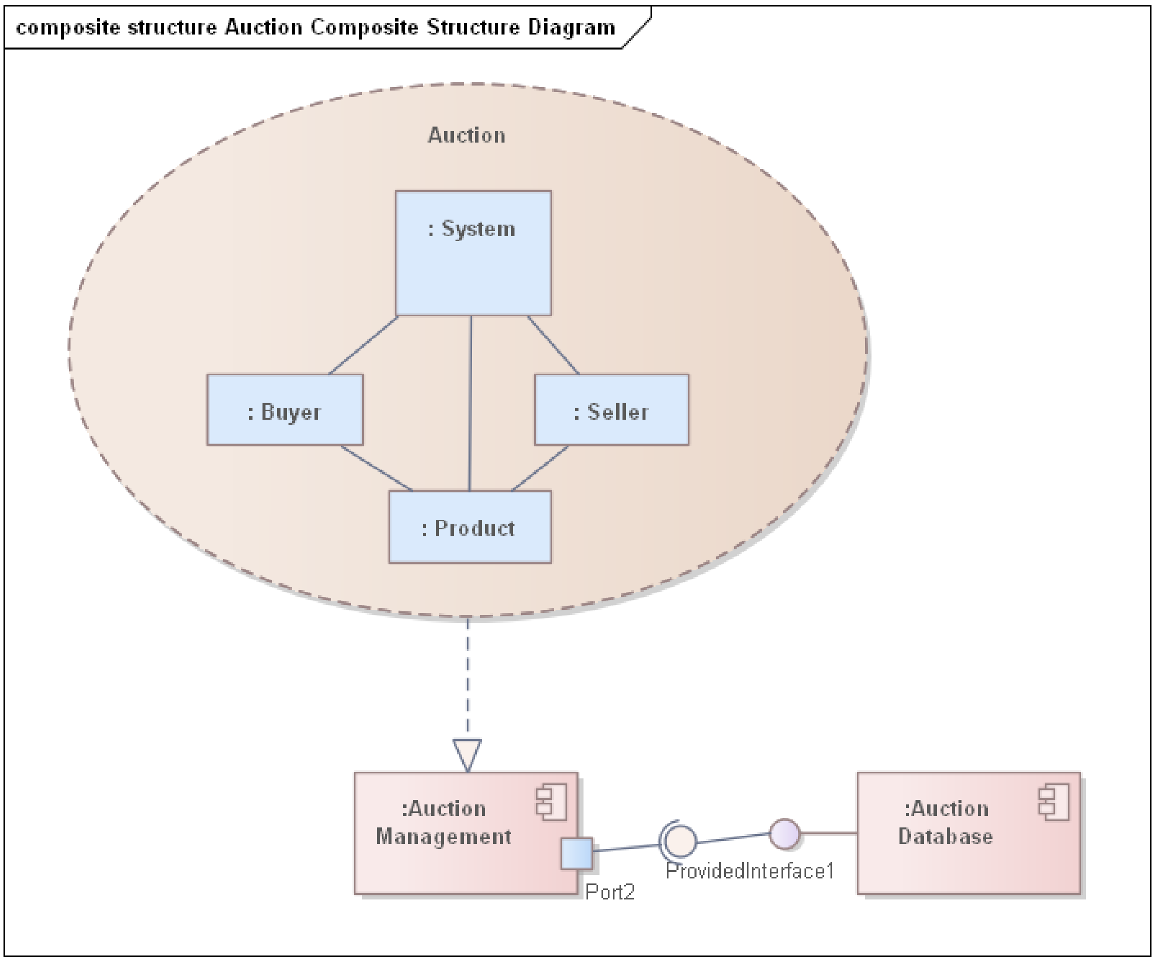


5 Komponenty / Components

5.1 Component diagram



5.2 Composite structure diagram



6 Kód / Code

In this part I implemented use cases Create an Auction and Attend an Auction in language Python. Here is the code:

```
import json


class User:
    def __init__(self, email, name, username):
        self.email = email
        self.name = name
        self.username = username
        self.address = None
        self.paymentInfo = []

    def addAddress(self, address):
        self.address = address

    def addPaymentInfo(self, payment):
        self.paymentInfo.append(payment)

    def toJSON(self):
        return json.dumps(self, default=lambda o: o.__dict__, sort_keys=True,
indent=4)


class Seller(User):
    def __init__(self, email, name, username):
        super().__init__(email, name, username)
        self.auctions = []

    def addAuction(self, auction):
        self.auctions.append(auction)

    def toJSON(self):
        return super().toJSON()


class Buyer(User):
    def __init__(self, email, name, username):
        super().__init__(email, name, username)

    def toJSON(self):
        return super().toJSON()


class Product:
    def __init__(self, name, price):
        self.name = name
        self.price = price

    def toJSON(self):
        return json.dumps(self, default=lambda o: o.__dict__, sort_keys=True,
indent=4)


class Coin(Product):
    def __init__(self, name, price, age, condition):
        super().__init__(name, price)
        self.age = age
        self.condition = condition

    def toJSON(self):
        return super().toJSON()


class CoinSet(Product):
    def __init__(self, name, price, coins, numberOfCoins, setName):
        super().__init__(name, price)
        self.coins = coins
        self.numberOfCoins = numberOfCoins
        self.setName = setName

    def toJSON(self):
        return super().toJSON()


class Address:
    def __init__(self, city, country, postalCode, street, state):
        self.city = city
        self.country = country
        self.postalCode = postalCode
        self.street = street
        self.state = state

    def toJSON(self):
        return json.dumps(self, default=lambda o: o.__dict__, sort_keys=True,
indent=4)


class Payment:
    def __init__(self, amount, paymentInfo):
        self.amount = amount
        self.paymentInfo = paymentInfo

    def toJSON(self):
        return json.dumps(self, default=lambda o: o.__dict__, sort_keys=True,
indent=4)


class PaymentInfo:
    def __init__(self, id):
        self.id = id

    def toJSON(self):
        return json.dumps(self, default=lambda o: o.__dict__, sort_keys=True,
indent=4)


class PayPalAccount(PaymentInfo):
    def __init__(self, id, token, username):
        super().__init__(id)
        self.token = token
        self.username = username

    def toJSON(self):
        return super().toJSON()


class CreditCard(PaymentInfo):
    def __init__(self, id, cardNumber, cardholderName, expirationDate,
securityCode):
        super().__init__(id)
        self.cardNumber = cardNumber
        self.cardholderName = cardholderName
        self.expirationDate = expirationDate
```

```python
        self.securityCode = securityCode

    def toJSON(self):
        return super().toJSON()


class Auction:
    def __init__(self, currentPrice, endTime, item, nextMinPrice, seller):
        self.currentPrice = currentPrice
        self.endTime = endTime
        self.item = item
        self.nextMinPrice = nextMinPrice
        self.seller = seller

    def toJSON(self):
        return json.dumps(self, default=lambda o: o.__dict__, sort_keys=True,
indent=4)


def sell():
    options = ("coin", "coinset")

    print("Select an option: ")
    for index, option in enumerate(options):
        print(index + 1, option)
    option = options[int(input("Enter the name of the option: ")) - 1]

    if option == "coin":
        # get all info for coin
        coinName = input("Enter the name of the coin: ")
        coinPrice = int(input("Enter the price of the coin: "))
        coinAge = int(input("Enter the age of the coin: "))
        coinCondition = input("Enter the condition of the coin: ")

        # create coin
        coin = Coin(coinName, coinPrice, coinAge, coinCondition)

        # create auction
        auction = Auction(
            coinPrice,
            "",
            coin,
            100,
            seller,
        )

        # add auction to seller
        seller.addAuction(auction)

        # add auction to auctions
        auctions.append(auction)
        print("Coin added to auction")
        # print coin info
        print("Coin Name: ", coin.name)
        print("Coin Price: ", coin.price)
        print("Coin Age: ", coin.age)
        print("Coin Condition: ", coin.condition)
        print("Price: ", auction.currentPrice)

        return
    elif option == "coinset":
        # get all info for coin set
        coinSetName = input("Enter the name of the coin set: ")
        coinSetPrice = int(input("Enter the price of the coin set: "))
        coinSetNumberOfCoins = int(input("Enter the number of coins in the coin
set: "))
        coinSetSetName = input("Enter the name of the set: ")

        # get coins for coin set
        coins = []
        for i in range(coinSetNumberOfCoins):
            coinName = input("Enter the name of the coin: ")
            coinPrice = int(input("Enter the price of the coin: "))
            coinAge = int(input("Enter the age of the coin: "))
            coinCondition = input("Enter the condition of the coin: ")

            coin = Coin(coinName, coinPrice, coinAge, coinCondition)
            coins.append(coin)

        # create coin set
        coinSet = CoinSet(
            coinSetName,
            coinSetPrice,
            coins,
            coinSetNumberOfCoins,
            coinSetSetName,
        )

        # create auction
        auction = Auction(
            coinSetPrice,
            "",
            coinSet,
            100,
            seller,
        )

        # add auction to seller
        seller.addAuction(auction)

        # add auction to auctions
        auctions.append(auction)
        print("Coin set created!")
        print("Coin set: " + coinSet.name)
        print("Price: " + str(coinSet.price))
        print("Number of coins: " + str(coinSet.numberOfCoins))
        print("Set name: " + coinSet.setName)
        print("Coins: ")
        for coin in coins:
            print(coin.name)

        return
    else:
        print("Invalid option")


def buy():
    print("Select an auction to attend: ")
    for index, auction in enumerate(auctions):
        print(index + 1, auction.item.name)
    auction = auctions[int(input("Enter the name of the auction: ")) - 1]

    print(
        f"Select bid amount (minimum: {auction.currentPrice +
auction.nextMinPrice}): "
    )

    bidAmount = int(input())

    if bidAmount < auction.currentPrice + auction.nextMinPrice:
        print("Bid amount is too low")
        return

    print("Congratulations! You won the auction!")

    print("Select payment method: ")
    for index, payment in enumerate(auction.seller.paymentInfo):
        print(index + 1, payment.cardNumber)
    payment = auction.seller.paymentInfo[
        int(input("Enter the name of the payment method: ")) - 1
    ]

    print("Thank you for your purchase!")


def main():
    option = input("Do you want to sell or buy? ")

    if option == "sell":
        sell()
    elif option == "buy":
        buy()
    else:
        print("Invalid option")
```

```
address = Address("New York", "USA", "10001", "Wall Street", "NY")
creditCard = CreditCard("1", "123456789", "John Doe", "12/20", "123")

buyer = Buyer("janedoe@test.test", "Jane Doe", "janedoe")
buyer.addAddress(address)
buyer.addPaymentInfo(creditCard)

seller = Seller("johndeer@test.test", "John Deer", "johndeer")
seller.addAddress(address)
seller.addPaymentInfo(creditCard)

auctions = [
    Auction(100, "12/12/12", Coin("Gold", 100, "12/12/12", "Good"), 100, seller),
    Auction(200, "12/12/12", Coin("Silver", 200, "12/12/12", "Good"), 100,
seller),
    Auction(300, "12/12/12", Coin("Platinum", 300, "12/12/12", "Good"), 100,
seller),
]

if __name__ == "__main__":
    main()
```

7 Algebraická špecifikácia / Algebraic specification

Types:

Auction, Product

Functions:

createAuction : Product → Auction
cancelAuction : Auction → Empty
updateAuctionProduct : Auction x Product → Auction
resolveAuction : Auction → Auction
exists : Auction → Boolean

Axioms:

∀a : Auction, p1 : Product, p2 : Product

A1 : cancelAuction(createAuction(p1)) = Empty
A2 : updateAuctionProduct(createAuction(p1), p2) = createAuction(p2)
A3 : exists(createAuction(p1))

Prequisites:

cancelAuction(a : Auction) requires exists(a : Auction)
updateAuctionProduct (a : Auction) requires exists(a : Auction)
resolveAuction (a : Auction) requires exists(a : Auction)

Note: Function exists(a : Auction) tells if auction object exsist or if it is already terminared/finished
Note 2: Functions are used in Auction controller

8 Model vlastností (nepovinná časť) / Property model (optional part)