

EECS 1022 Group Project

Group Name: AACM Studios

Name	Email	Section + Lab
Adam Stolnits	adamstol@my.yorku.ca	Section N Lab 1
Ajay Bukkaraya	ajaybn@my.yorku.ca	Section N Lab 2
Chenyang Liang	harryl@my.yorku.ca	Section N Lab 4
Mickey Byalsky	mickeyb@my.yorku.ca	Section N Lab 2

Title of Project: NFT Pioneers (NFTP)

GitHub link: <https://github.com/WTIOFlameburst/NFTPioneers>

Description: Our project is an app that simulates NFT trading. The user's goal is to make \$5000 in the least amount of "days" possible. The program will contain databases (arrays) of each NFT object. Images, titles, and prices are all part of each individual NFT object. The NFTs can be sorted in a number of ways, including alphabetical, price, and type. A button allows the user to update NFT prices as they vary randomly, much like real-world prices. On the profile screen, the user can keep track of the worth of their account. When a user's market worth exceeds \$5000, the app shows how many "days" it took them to get there.

Functional Requirements:

- ☐ Sorting algorithm to sort by the following categories: Alphabetical, Price High to Low, Price Low to High, and Sub-Category of NFT (I.e, Space Monkeys, Crypto Kitties, etc.)
- ☐ Buy NFT's
 - User will be able to buy NFTs
- ☐ Sell NFTs
 - User will be able to sell NFTs
- ☐ Account Login
 - There is account creation functionality
- ☐ Next Day Button (Refresh)
 - Press a button that calls a method that updates all prices of NFT's
 - Using a Random number generator takes a number within a predefined range and can increase or decrease by this range. (Ex. NFT1 Price (0-200, 0-1) where 0-200 is the amount the price changes and where 0-1 indicates either negative (0) or positive (1)).
- ☐ Base Money

- User starts off with \$1000
- ☐ Add Money
 - User is able to add money to their trading account to purchase more NFT's
- ☐ Withdraw Money
 - User is able to withdraw money to their bank account
- ☐ Current NFT Holdings
 - Page that shows what NFT's are held by the user.
- ☐ NFT Store Page
 - Page that shows the user what NFT's are for sale
- ☐ Total Balance Tracker
 - Including Cash and NFT market value(s)
- ☐ NFT Description
 - About the NFT
- ☐ Colour Changing text
 - If the price of NFT for the day went up, text will become green, other red text colour if negative.
- ☐ Instructions page
 - Short instructions page educating the user on NFT trading (buy low sell high, etc.)

Phase 3: Deadline 1:

So far our group has been mainly focused on coding the project in IntelliJ. No implementation has been done yet using Android Studio however the coding process has been completed.

Coding progress can be seen on github:

<https://github.com/WTIOFlameburst/NFTPioneers.git>

Pseudo Code can be seen below

Of our functional requirements above we have completed coding:

Buy NFTs

Sell NFTs

Next Day Button Refresh

Base Money

Total Balance Tracker

Instructions Page

Eight sorting classes to sort the NFTs lists by (ex. By name, category, rarity, price, and the previous terms but in descending order)

As well as new functional requirements that have not been mentioned above such as:

NFT Object Class

List of NFTs Object Class

Seven unique test cases for our NFT class and sorting methods

Add NFTs

While working on the project we've decided to change and remove a few functional requirements to better suit our project:

~~Add Money~~ ⇒ Decided to remove the add money functionality to make the game more difficult and relies on the user to take more risks when playing.

~~Withdraw Money~~ ⇒ Automatically adds profits to current balance

- Withdrawing money collected from NFTs added too many extra steps for the user and would be quite tedious for the player to always have to withdraw money from their balance to their bank account. Instead the current balance will be treated as the player's bank account.

Goals for the following two weeks / Phase 3 Deadline 2:

Complete the front end portion of our project in Android Studio (**Ajay and Mickey**)

Incorporate the code from java into the project in Android Studio (**Adam and Chen**)

Create more test cases to prevent and check for any flaws/bugs in our code (**Chen**)

Logo design and NFT image design (**Adam**)

Design of Classes / Pseudo Code:

Algorithm: nft

Pre-Conditions: Valid inputs for String name, String category, String imageName, String rarity, int priceOfNFT

Post-Conditions: Formatted String Output ("Name (%s), Category (%s), Image Name (%s), Rarity (%s), Current Price: (%.2f)", name, category, imageName, rarity, priceOfNFT));

```
name ← String
category ← String
imageName ← String
rarity ← String
priceOfNFT ← int
```

Constructor nft

nft()

Constructor nft

Pre-Conditions: Valid inputs for String name, String category, String imageName, String rarity, int priceOfNFT

Post-Conditions: Global variables set to correct inputs from parameters

```
nft(String name, String category, String imageName, String rarity, int priceOfNFT)
    this.name ← name
    this.category ← category
    this.imageName ← imageName
    this.rarity ← rarity
    this.priceOfNFT ← priceOfNFT
```

Method changePrice

Pre-Conditions: Valid inputs for double currentPrice and String rarity

Post-Conditions: Sets priceOfNFT to correct price

```
changePrice ← 0, int
If (rarity = "Common") then
    changePrice ← (int)(Math.random() * 50 + 1)
Else if (rarity = "Epic") then
    changePrice ← (int)(Math.random() * 250 + 50)
Else if (rarity = "Legendary") then
    changePrice ← (int)(Math.random() * 500 + 250);
upOrDown ← (Math.random() * 2 + 1), int
If (upOrDown = 1) then
    setPriceOfNFT((int) currentPrice + changePrice)
Else if (upOrDown = 2) then
    setPriceOfNFT((int) currentPrice - changePrice)
    If ((int) currentPrice - changePrice < 0)
        setPriceOfNFT(0)
```

Method getName

Pre-Conditions: None Specified

Post-Conditions: Returns name

return name

Method getCategory:

Pre-Conditions: None Specified

Post-Conditions: Returns category

return category

Method getImageName:

Pre-Conditions: None Specified

Post-Conditions: Returns imageName

return imageName

Method getRarity:

Pre-Conditions: None Specified

Post-Conditions: Returns Rarity

return rarity

Method getPriceOfNFT:

Pre-Conditions: None Specified

Post-Conditions: Returns priceOfNFT

return priceOfNFT

Method setName:

Pre-Conditions: String name

Post-Conditions: this.name = name

this.name ← name

Method setCategory:

Pre-Conditions: String category

Post-Conditions: this.category = category

this.category ← category

Method setImageName:

Pre-Conditions: String imageName

Post-Conditions: this.imageName = imageName

this.imageName ← imageName

Method setRarity:

Pre-Conditions: String rarity

Post-Conditions: this.rarity = rarity

this.rarity ← rarity

Method setPriceOfNFT:

Pre-Conditions: int priceOfNFT

Post-Conditions: this.priceOfNft = priceOfNFT

this.priceOfNFT \leftarrow priceOfNFT

Method getNFTValues:

Pre-Conditions: None Specified

Post-Conditions: Formatted String Output

String getNFTValues()

return Formatted String ("Name (%s), Category (%s), Image Name (%s), Rarity (%s),
Current Price: (%.2f)", name, category, imageName, rarity, priceOfNFT))

Algorithm: ListOfNFTS

import java.util.ArrayList

import java.util.Collections

ArrayList nftList \leftarrow new ArrayList(), ArrayList

Constructor ListOfNFTS

Pre-Conditions: None Specified

Post-Conditions: None Specified

ListOfNFTS()

Constructor ListOfNFTS

Pre-Conditions: ArrayList nftList

Post-Conditions: this.nftList = nftList

ListOfNFTS(ArrayList nftList)

this.nftList \leftarrow nftList

Method sortNFT

Pre-Conditions: ArrayList nftList and int choose

Post-Conditions: Correctly sorts nftList

If (choose = 1) then

Collections.sort(nftList, new nftSortName())

Else if (choose = 2) then

Collections.sort(nftList, new nftSortNameDescending())

Else if (choose = 3) then

Collections.sort(nftList, new nftSortPrice())

Else if (choose = 4) then

Collections.sort(nftList, new nftSortPriceDescending())

Else if (choose = 5) then

Collections.sort(nftList, new nftSortRarity())

Else if (choose = 6) then

```
    Collections.sort(nftList, new nftSortRarityDescending())
Else if (choose = 7) then
    Collections.sort(nftList, new nftSortCategory())
Else
    Collections.sort(nftList, new nftSortCategoryDescending())
```

Method setNFTList

Pre-Conditions: ArrayList nftList

Post-Conditions: this.nftList = nftList

this.nftList ← nftList

Method getArrayList:

Pre-Conditions: None Specified

Post-Conditions: ArrayList nftList

return nftList

Method addNFT:

Pre-Conditions: nft nft

Post-Conditions: adds nft to nftList

nftList.add(nft)

Algorithm: nftSortCategory

import java.util.Comparator;

Method compare

Pre-Conditions: nft o1 and nft o2

Post-Conditions: Returns an int

int compare(nft o1, nft o2)

return o1.getCategory().compareTo(o2.getCategory())

Algorithm: nftSortCategoryDescending

Method compare

Pre-Conditions: nft o1 and nft o2

Post-Conditions: Returns an int

int compare(nft o1, nft o2)

return o2.getCategory().compareTo(o1.getCategory())

Algorithm: nftSortName

Method compare

Pre-Conditions: nft o1 and nft o2

Post-Conditions: Returns an int

int compare(nft o1, nft o2)

return o1.getName().compareTo(o2.getName())

Algorithm: nftSortNameDescending

Method compare

Pre-Conditions: nft o1 and nft o2

Post-Conditions: Returns an int

int compare(nft o1, nft o2)

return o2.getName().compareTo(o1.getName())

Algorithm: nftSortRarity

Method compare

Pre-Conditions: nft o1 and nft o2

Post-Conditions: Returns an int

int compare(nft o1, nft o2)

return o1.getRarity().compareTo(o2.getRarity())

Algorithm: nftSortRarityDescending

Method compare

Pre-Conditions: nft o1 and nft o2

Post-Conditions: Returns an int

int compare(nft o1, nft o2)

return o2.getRarity().compareTo(o1.getRarity());

Algorithm: nftSortPrice

Method compare

Pre-Conditions: nft o1 and nft o2

Post-Conditions: Returns an int

int compare(nft o1, nft o2)


```
return o1.getPriceOfNFT() - (o2.getPriceOfNFT())
```

Algorithm: nftSortPriceDescending

Method compare

Pre-Conditions: nft o1 and nft o2

Post-Conditions: Returns an int

int compare(nft o1, nft o2)

```
    return o2.getPriceOfNFT() - (o1.getPriceOfNFT())
```