

Member ID: _____

Time: _____

Rank: _____



JAVA PROGRAMMING

(340)

REGIONAL 2023

PRODUCTION:

NFT Market Place

_____ (425 points)

Test Time: 90 minutes

GENERAL GUIDELINES:

Failure to adhere to any of the following rules will result in disqualification:

1. Member must hand in this test booklet and all printouts if any. Failure to do so will result in disqualification.
2. No equipment, supplies, or materials other than those specified for this event are allowed in the testing area. No previous BPA tests and/or sample tests (handwritten, photocopied, or keyed) are allowed in the testing area.
3. Electronic devices will be monitored according to ACT standards.

You will have ninety (90) minutes to complete your work.

Your name and/or school name should *not* appear on work you submit for grading.

1. Create a folder on the flash drive provided using your contestant number as the name of the folder.
2. Copy your entire solution/project into this folder.
3. Submit your entire solution/project so that the graders may open your project to review the source code.
4. Ensure that the files required to run your program are present and will execute on the flash drive provided.

*Note that the flash drive letter may *not* be the same when the program is graded as it was when you created the program.

*It is recommended that you use relative paths rather than absolute paths to ensure that the program will run regardless of the flash drive letter. It is **HIGHLY** recommended that you place all of the files into one folder. Advanced IDE's will place your files into multiple folders that are noncentralized which could have catastrophic effects on your program during grading on third party machines. Graders will not make any adjustments to your program.

The graders will *not* compile or alter your source code to correct for this.
Submissions that do *not* contain source code will *not* be graded.

Assumptions to make when taking this assessment:

- There will only be one record (object) created from the supplied data fields in the comments section.
- Users can attempt to enter in erroneous information during an input prompt.
- The program will only exit at a single point which is when they decline to retrieve the single record created.
- All getter and setter methods for the NFTData and WalletAddress object have been created; the source code for these classes is not accessible.
- If the user does not want to create an NFT, the program will ask if they want to retrieve the records. If they answer "no" the program terminates. If they answer yes then the output (supplied in this document) will have NO information associated with the NFT such as its address, floor price, and rarity.

Development Standards:

- Your Code must use a consistent variable naming convention.
- All subroutines (if any), functions (if any), and methods (if any) must be documented with comments explaining the purpose of the method, the input parameters (if any), and the output (if any). Readability is a goal of good code.
- If you create a class, then you must use Javadoc comments.

NFT Market Place

In this challenge you will be required to create a software program that will allow a company to begin the initial stages of programming a software that will take user information to create an account for a new client and allow that client to create their own NFT. NFT is a non-fungible token and they are pieces of digital contents linked together on the blockchain. Most often NFT's are used to store digital artwork and other media files but they can store anything that is a computer file or group of files. In this challenge you will be allowing the user to create their own NFT, not in the sense of uploading a digital asset, but instead of creating the address which is the location of the NFT on the block chain. A blockchain is a distributed database or network of computers that is considered to be highly decentralized. It is most often associated with Bitcoin Ethereum projects. You do not need to have any understanding of blockchains or NFTs to complete this challenge.

The NFT Market Place challenge will require you to use given information to create a client object that will store their username and their NFT project name. Next, you will give the user the option to create a new NFT. When they create their new NFT they will be required to give it a floor price which essentially is the beginning price it will have when enters a marketplace. You will not be creating the marketplace. After the floor price has been set you will next declare what type of NFT it will be. You will give the user two choices of NFT's of being an ERC-721 token or a BEP-721 token. After choosing which token type, your program will automatically and randomly generate an NFT address which will be explained later. After the address has been created the user will be required to give the NFT a rarity level that ranges from "Common" that has a value of 1 all the way to "Mythic" which has a value of 6. Finally, the program will conclude with allowing the user to see the new record that was created. The NFT address that is randomly created will have the following criteria:

- The first three alpha-numeric characters of the address will be the first three alpha-numeric values from the username. For example, if the username is Ana1234, then the first three values for the random address will be "Ana".
- Next you will concatenate four randomly generated set of numbers that range from 0000 to 9999. Each set of randomly generated numbers be separated with the following delimiter: 'x'. It does not matter if there are two x's beside each other.
- The final step of the address will entail concatenating which type of NFT token was selected. For example, if an ERC-721 token was selected when creating the NFT, then that token type will be the value concatenated at the end of the address. The hyphens will need to be removed.
- Here is an example of an NFT created by Ana123 who chose an ERC-721 token type:
 - Anax5180xx1912xx1767xx5653xERC721
 - Note: the 'x' delimiter is used to separate four sets of random numbers.
 - The random numbers must have four digits. A random value of 43 would need to be 0043.

Input

After the program creates the NFTData object that stores the user information the program will prompt the user if they would like to create a new NFT. After selecting yes, the program will next prompt them to create a floor price for the NFT with a value that ranges from \$0.50 to \$99,999.99. Third, the user will be prompted to enter in a '1' (ERC-721) or '2' (BEP-721) to decide which type of NFT token they want to create. Fourth, the program will prompt the user to select the rarity level of the NFT by entering in a numeric value ((6) Mythic, (5) Legendary, (4) Epic, (3) Rare, (2) Uncommon, (1) Common) for the six types. Finally, the user will be prompted to see the profile just created with a yes or no, which a selection of yes will display the record. A "no" answer will terminate the program. It should also be noted, that the program should accept an upper- or lower-case spelling for yes or no, as well as allow the user to enter in a 'y' or 'Y' for "yes", and a 'n' or 'N' for "no".

Output

The first output to the console will be indicating to the user that their object was successfully created per the requirements for the NFTData constructor. The username is displayed in the output (Ana1234).

Welcome Ana1234 to the NFT Market Place form. Your wallet is now connected.

Type in "Yes" if you want create a new NFT:

ENTER: Yes or No:

Based upon the user entering in 'y' or "yes", they will be prompted with creating a floor price:

All new NFT's require a new floor price. How much will be the NFT's floor price?

Please enter in a value between \$0.50 to \$99,999.99:

After entering an appropriate number in the given range, the user will next be prompted to enter a 1 or 2 for which type of token the NFT will be. Based upon which token was selected, the program will automatically generate the randomized NFT contract address:

Please enter a '1' for a ERC-721 token, or enter '2' for a BEP-721 token: 1

Your NFT contract address: Anax5477xx4525xx2451xx2101xERC721

The user will be prompted to choose what type of rarity level the NFT will possess.

All new NFT's require a rarity. Choose the number for the level of rarity:

(6)Mythic, (5)Legendary, (4)Epic, (3)Rare, (2)Uncommon, (1)Common

Please enter in a value between 1 to 6:

Finally, the program will ask the user if they would like to see the profile for the new user (NOTE: this step will also terminate the program):

Do you want to see the profile for this user?

ENTER: Yes or No: y

```
*****
***** CREATOR RECORDS *****
```

Username: Ana1234

Wallet Address: 1Awyd1QWR5gcfrn1UmL8dUBj2H1eVKtQhg

Collection Name: Bored BPA Yacht Club

NFT Address: Anax5477xx4525xx2451xx2101xERC721

Floor Price: \$456.56

Rarity/Collection: Common

```
*****
*****
```

Sample Run Examples (NOTE: these examples have formatting features such **bold** and *italicized* fonts to bring your attention to key entry actions by the user; your program is not required to display such formatting)

Sample Run #1.a with Update Entry Error (repeats the entry request until 'n','y', "Yes" or "No" entered)

Welcome Ana1234 to the NFT Market Place form. Your wallet is now connected.

Type in "Yes" if you want create a new NFT:

ENTER: Yes or No: *t*

Type in "Yes" if you want create a new NFT:

ENTER: Yes or No:

Sample Run #1.b with Retrieval Entry Error (repeats entry request until 'n','y', "Yes" or "No" entered); notice that the Creator Records will omit all of the NFT information.

Welcome Ana1234 to the NFT Market Place form. Your wallet is now connected.

Type in "Yes" if you want create a new NFT:

ENTER: Yes or No: *n*

Do you want to see the profile for this user?

ENTER: Yes or No: *ewrt*

Do you want to see the profile for this user?

ENTER: Yes or No: *y*

***** CREATOR RECORDS *****

Username: Ana1234

Wallet Address: 1Awyd1QWR5gcfrn1UmL8dUBj2H1eVKtQhg

Collection Name: Bored BPA Yacht Club

Sample Run #2 with Value and Range Entry Error for Floor Price setting. (NOTE: the value entry error will prompt a message whereas a range entry error will just keep prompting the user to enter in the correct value which is caused by two different entry error detections (one for value and the other for range).

All new NFT's require a new Floor Price. How much will be the NFT's floor price?

Please enter in a value between \$0.50 to \$99,999.99: *wrong*

Please enter a correct value.

Please enter in a value between \$0.50 to \$99,999.99:

RANGE ERRORS (low and high)

Please enter in a value between \$0.50 to \$99,999.99: *0*

Please enter in a value between \$0.50 to \$99,999.99:

Please enter in a value between \$0.50 to \$99,999.99: *100000*

Please enter in a value between \$0.50 to \$99,999.99:

Sample Run #3 with Value and Range Entry Error for Token Type setting. NOTE: the value entry error will prompt a warning message whereas a range entry error will just keep prompting the user to enter in the correct value.

Please enter in a 1 for ERC-721 or a 2 for a BEP-721 token type: *f*

Please enter a correct value.

Please enter in a 1 for ERC-721 or a 2 for a BEP-721 token type: *e*

Please enter a correct value.

Please enter in a 1 for ERC-721 or a 2 for a BEP-721 token type: *4*

Please enter in a 1 for ERC-721 or a 2 for a BEP-721 token type:

Sample Run #4 with Value and Range Entry Error for Rarity setting. NOTE: the value entry error will prompt a warning message whereas a range entry error will just keep prompting the user to enter in the correct value.

All new NFT's require a rarity. Choose the number for the level of rarity:

(6)Mythic, (5)Legendary, (4)Epic, (3)Rare, (2)Uncommon, (1)Common

Please enter in a value between 1 to 6: *f*

Please enter a correct value.

Please enter in a value between 1 to 6: *sadfgd*

Please enter a correct value.

Please enter in a value between 1 to 6: *0*

Please enter in a value between 1 to 6: *9*

Sample Run #5 user enters “no” or ‘n’ when prompted by the program, thus terminating the program (entry errors will cause the program to keep prompting):

Do you want to see the profile for this user?

ENTER: Yes or No: *n*

Have a great day and see you on the Moon!

Sample Run #6 if user enters creates the NFT properly and chooses to view the profile;

NOTE: the NFT info appears:

```
*****
*****
***** CREATOR RECORDS *****
Username: Ana1234
Wallet Address: 1Awyd1QWR5gcfrn1UmL8dUBj2H1eVKtQhg
Collection Name: Bored BPA Yacht Club
NFT Address: Anax7061xx7769xx1404xx1748xBEP721
Floor Price: $456.00
Rarity/Collection: Rare
*****
*****
```

Requirements:

1. You must create an application with the main class called NFTMarketPlace. This class will create the NFTData object and its nested attribute which is the WalletAddress object. The NFTData and WalletAddress classes are already created and are provided in your resource files.
2. The NFTMarketPlace will be the driver program that runs the data entry prompts from the user, formats data, and sets/gets information from the NFTData object.
3. Your contestant number must appear as a comment at the top of the main source code file.
4. If incorrect data is entered, then the program should display an appropriate/similar message as shown in the Sample Runs above. The program only exits when the user chooses to not view the profile or chooses to view the profile.
5. The program will display the outputs as shown above examples above.

Solution and Project		
The project is present on the flash drive	_____	10 points
The projects main class is named NFTMarketPlace	_____	10 points
The class helper method is named setFloorPriceNFTDataRecord(NFTData c)	_____	10 points
The class helper method is named setNFTRarityRecord(NFTData c)	_____	10 points
The class helper method is named setContractAddress(NFTData trader)	_____	10 points
The class helper method is named getUserStringInput()	_____	10 points
The class helper method is named getUserNumericalInput(double low, double high, String message)	_____	10 points
The class helper method is named consoleRecordCheck(NFTData c)	_____	10 points
Program Execution		
The program runs from the USB flash drive	_____	15 points
<i>If the program does not execute, then the remaining items in this section receive a score of zero.</i>		
The program displays a message welcoming with UserName and confirming the wallet is now connected.	_____	10 points
The program prompts and forces user to enter “yes” or “no” if they want to create a new NFT.	_____	10 points
If “no” or ‘n’ is entered for update: the program prompts and forces user to enter “yes”, ‘y’, ‘n’, or “no” if they want to view the profile for the user.	_____	10 points
If “yes” or ‘y’ is entered for update: the program prompts and forces user to enter “\$0.50 to \$99,999.99” the deposit amount. All data entry errors are caught.	_____	10 points
Program displays “Creator Records”(aka profile) with NFT information when appropriate, no NFT information when appropriate.	_____	30 points
NFT Address is a combination between first three letters of username, randomly generated four sets of 4 digits that use the delimiter of ‘x’ (‘xx’ is permissible). The Address has the non hyphenated token type concatenated at the end . i.e. NFT Address: <i>Anax7061xx7769xx1404xx1748xBEP721</i>	_____	30 points
The program prompts and forces user to enter “yes: or “no” if they want to view the profile & it is not case sensitive.	_____	10 points
If “no” or ‘n’ is entered for retrieval: the program prompts says “Have a great day and see you on the Moon!” and then terminates.	_____	10 points
Floor price is displayed for US Dollars	_____	10 points
Output matches required format.	_____	20 points

Source Code Review		
The source code is properly commented		
A comment containing the contestant number is present	_____	10 points
Methods and code sections are commented	_____	20 points
Code uses try... catch for exception handling for getUserNumericalInput(double, double, String) when entering numbers. All values entered beyond given range are not accepted based upon high and low parameters; and all rational values	_____	30 points
All rational numbers entered via getUserNumericalInput(double, double, String) are truncated when being passed back to integer variable (NOTE: easiest method is to cast the returned value)	_____	10 points
Appropriate messages appear (3 total: floor price double, token type int, rarity type int) via getUserNumericalInput(double, double, String) via passed String	_____	10 points
Code uses getUserStringInput() to gather all “yes”, “no”, ‘n’ or ‘y’ entries and passes result back as String.	_____	10 points
main (String args []) : NFTData object is correctly constructed from data entry into its given attributes, and also properly passes data into attribute of WalletAddress object (attribute in NFTData).	_____	10 points
main (String args []) : program checks to see if NFTData object and the WalletAddress have been created. Displays a welcome message with getting the Username and the wallet address from the getter methods	_____	10 points
setContractAddress(NFTData trader) : method creates the NFT address based upon the given criteria (first three letters of username, four sets of 4 random digits separated by ‘x’ delimiter, non hyphenated token type concatenated at the end)	_____	30 points
consoleRecordCheck(NFTData c) : formats floor price to US dollars using NumberFormat	_____	20 points
consoleRecordCheck(NFTData c) : prints all required data via accessing getter methods of the NFTData object.	_____	20 points
Total Points		/425 points

Class Behaviors and Attributes: Help Information

(NOTE: this is the class you will program)

public class NFTMarketPlace Attributes (NOTE: all are provided):

- public static String rarity []: stores the rarity levels
- private static double lowFloor: lower range value for the Floor Price
- private static double highFloor: upper range value for the Floor Price
- private static String floorMssg: message prompting user for Floor Price entry
- private static int lowRarity: lower range value for the Rarity type
- private static int highRarity: upper range value for the Rarity type
- private static String rarityMssg: message prompting user for Rarity type entry
- private static int lowToken: lower range value for the Token type
- private static int highToken: upper range value for the Token type
- private static String tokenMssg: message prompting user for Token type
- static Scanner sc: Scanner object for user entry

public class NFTMarketPlace Behaviors:

- PROVIDED: public static void main (String args []): a combination of processes that you will create and others that have been provided for you. Your task here is to create the objects, check they are not null and print out the welcome message with the proper object method calls. Other processes will initiate the program to begin calling support methods that you will be in charge of creating.
- PROVIDED: private static void setFloorPriceNFTDataRecord(NFTData c): this method is provided. Gets user input via another method you will create, sets the price in the object, and then sets the contract address via another method you will create. NOTE: this method is called from the Main method and is provided.
- PROVIDED: private static void setNFTRarityRecord(NFTData c): this method is provided. Gets user input via another method you will create, sets the rarity level in the object. NOTE: this method is called from the Main method and is provided.
- ACTION: private static void setContractAddress(NFTData trader): you will create this method by creating the contract address based upon the requirements. The object will need to have address value set to it by calling its object method setNFTAddress(). In addition, this method will also print to console the NFT address once it is created.
- ACTION: private static String getUserStringInput(): you will create this method by getting the user input for a yes/no question prompts. NOTE: this method is called from the given processes located in the Main method.
- ACTION: private static double getUserNumericalInput(double low, double high, String message): you will create this flexible method to get any numerical entry for ranges of values. The parameters should be used to set low/high ranges and have the message prompt for the user. Your method must be able to handle any input mismatch exception errors and force the user to enter in correct values. NOTE: the return value is a double; however, the return value is cast to an integer for the rarity and token type. The rarity has

already been programmed in the setNFTRarityRecord(NFTData c) method. You will need to program this action when creating the Token type. It is recommended, but not required, to have the token type set during the NFT contract address creation in the method setContractAddress(NFTData trader).

- ACTION: private static void consoleRecordCheck(NFTData c): you will create this method to print out the new NFT record created through the other methods. This is the final step. You will need to format the floor price to be in US dollars with proper notation and all expected place values. For example a price of \$5.50 should include the '0' in the hundredths place as opposed to being printed as \$5.5. You will also need to call the object methods to get the appropriate information to print to console. It is also important that this method is flexible in the situation the user does not create an NFT then the items printed will exclude any NFT related items. Likewise, if there is an NFT created this will be included.

class NFTData Behaviors and Attribute (NOTE: all are provided and you will only have access to the .class files):

Constructor:

- public NFTData (String userName, String collectionName)

Attributes:

- private String userName;
- private String productName;
- private String rarity;
- private String collectionName;
- private double floorPrice;
- private String tokenType;
- private String nftAddress = null;
- public WalletAddress walletAddress;

Behaviors

- public String getUN(): gets username
- public void setUN(String un): sets username
- public String getPN(): returns product name
- public void setPN(String pn): sets product name
- public String getRarity(): returns the rarity
- public void setRarity(String ra): sets rarity
- public String getCollection(): returns collection name
- public void setCollection(String co): sets collection name
- public Double getFloorPrice(): returns Floor Price
- public void setFloorPrice(Double f): sets Floor Price amount
- public String getTokenType(): returns Token Type

- `public void setTokenType(String bc):` sets Token Type
- `public void setNFTAddress(String a):` sets the NFT address
- `public String getNFTAddress():` returns NFT address

class WalletAddress Behaviors and Attribute (NOTE: all are provided and you will only have access to the .class files):

Constructor:

- `public WalletAddress()`
- `public WalletAddress(String wallet address)`

Attributes:

- `private String publicKeyWallet;`

Behaviors

- `public String getAdd():` returns the wallet address
- `public void setAdd(String s):` sets the wallet address