Shabari - Assessment - 2

---

1. Question No 1

Write a Java program that prompts the user to enter a secret key for a lock. The key must be an isogram, meaning no letter occurs more than once in the key. If the key throw a custom exception named NotAnIsogramException. The program should inform the user and ask for another secret key until a valid isogram is entered.

Analysis

- We need to get a secret lock key from the user
- We need to check whether the given string is Isogram
- If it is not an Isogram we nee to throw an Exception `NotAnIsogramException`.
- Otherwise print it is an Isogram

Code:

```java
package assessment;

import java.util.Scanner;

class NotAIsogramException extends Exception {
    NotAIsogramException(String str) {
        super(str);
    }
}

public class QuestionNo1 {
    public static boolean isIsogram(String key) throws NotAIsogramException {
        for (int i = 0; i < key.length(); i++) {
            for(int j = i + 1; j < key.length(); j++){
                if (key.charAt(i) == key.charAt(j))
                    throw new NotAIsogramException("The secret key is not an Isogram");
            }
        }
        return true;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        while (true) {
            System.out.print("Enter the Secret Key for the Lock: ");
            String key = sc.nextLine();
            try {
                if (isIsogram(key)) {
                    System.out.println("The secret key is an Isogram.");
                    break;
                }
            } catch (Exception e) {
                System.out.println(e.getMessage());
            }
        }
    }
}
```

Output

```
Enter the Secret Key for the Lock: shabari
The secret key is not an Isogram
Enter the Secret Key for the Lock: computer
The secret key is an Isogram.
```

---

2. Question No 2

A banking application that involves different types of accounts, such as savings accounts and checking accounts. Both types of accounts share a common attribute, wh
accounts have an additional specific attribute called the overdraft limit. To design the classes for these accounts we will define two interfaces: IBankAccount and IChec

The IBankAccount interface will have a property called Balance, which represents the account balance. It will be implemented by both the savings account and checki
ICheckingAccount interface will have a property called OverdraftLimit, which represents the maximum negative balance that a checking account can have. It will be im
account class.

Using these interfaces, we can achieve multiple inheritances by implementing both interfaces in the appropriate classes SavingAccount and CheckingAccount, and dis
transaction.

Analysis

- We need to get te input from the user.
- Counting the letters to check whether it is palindrome
- Return true if it is Palindrome
- Otherwise return false

Code:

```
package assessment;



interface IBankAccount{
    void setBalance(double balance);
    double getBalance();

    boolean deposit(double amount);

}

interface ICheckingAccount {
    double getOverdraftLimit();
    double getBalance();
    void setBalance(double balance);

    void withdraw(double amount) throws Exception;
}

class SavingsAccount implements IBankAccount{
    private double balance;

    public void display() {
        System.out.println("Savings Account: \n\t InitialBalance: " + getBalance());
    }

    @Override
    public void setBalance(double balance) {
        this.balance = balance;
    }
```

```
    @Override
    public double getBalance() {
        return balance;
    }

    @Override
    public boolean deposit(double amount) {
        try {
            this.balance += amount;
            return true;
        } catch (Exception e){
            return false;
        }
    }

}
class CheckingAccount implements ICheckingAccount{
    private double overdraftLimit = 1000;
    private double balance;
    @Override
    public double getBalance() {
        return this.balance;
    }

    @Override
    public void setBalance(double balance) {
        this.balance = balance;
    }

    @Override
    public void withdraw(double amount) throws Exception {
        if (amount > this.balance ){
            throw new Exception("\t Invalid Transaction: The Balance is less then the amount you entered.");
        } else if (this.overdraftLimit > (this.balance - amount)) {
            System.out.println("\t Overdraft Limit: " + getOverdraftLimit());
            throw new Exception("\t Invalid Transaction: You can't withdraw money less than OverdraftLimit");
        } else {
            this.balance -= amount;
        }
    }


    public void display() {
        System.out.println("Checking Account: \n\t InitialBalance: "+ getBalance()
                +"\n\t Overdraft Limit: " + getOverdraftLimit());
    }

    @Override
    public double getOverdraftLimit() {
        return this.overdraftLimit;
    }



}
public class QuestionNo2 {
    public static void main(String[] args) {
        SavingsAccount obj1 = new SavingsAccount();
        obj1.setBalance(2000);
        obj1.display();
        CheckingAccount obj2 = new CheckingAccount();
        obj2.setBalance(5000);
        obj2.display();
        System.out.println("Performing Transaction:");
        try {
```

```
            obj1.deposit(1000);
            System.out.println("\t The amount in your savings accout " + obj1.getBalance());
            obj2.withdraw(4500);
            System.out.println("\t The amount after withdraw in your checking account " + obj2.getBalance());
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
```

Output

```
Savings Account:
         InitialBalance: 2000.0
Checking Account:
         InitialBalance: 5000.0
         Overdraft Limit: 1000.0
Performing Transaction:
         The amount in your savings accout 3000.0
         Overdraft Limit: 1000.0
         Invalid Transaction: You can't withdraw money less than OverdraftLimit
```

---

3. Question No 3

Counting Good Pairs in Event Participant IDs (8 marks) You are organizing a large event where each participant is assigned a unique ID number. However, due to a regis ended up receiving the same ID. As a result, you have a list of participant IDs and you need to count how many pairs of participants have the same ID. These pairs are c good if the IDs are the same and i < j.Task is to Given a list of participant IDs, return the number of good pairs.

Analysis

- We need to get the event name from the user
- We need to get the array size and values from the user.
- We need to count the good pairs.
- Finally, we need to return the count.

Code

```
package assessment;

import java.util.ArrayList;
import java.util.Scanner;

public class QuestionNo3 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the event name: ");
        String event = sc.nextLine();
        System.out.print("Enter the size of array: ");
        int size = sc.nextInt();
        if (size <= 0){
            System.out.println("The size of array must be positive");
            return;
        }
        ArrayList<Integer> arr = new ArrayList<>();
        System.out.println("Enter the array Elements: ");
        for (int i = 0; i < size; i++) {
            arr.add(sc.nextInt());
        }
```

```
        int count = 0;
        for (int i = 0; i < size; i++) {
            for (int j = i + 1; j < size; j++) {
                if (arr.get(i) == arr.get(j))
                    count++;
            }
        }

        System.out.println("For the event "+ event+ " the count of good pair is " + count);
    }
}
```

Output

```
Enter the event name:
Tech Conference
Enter the size of array: 7
Enter the array Elements:
1 2 3 1 1 1 2
For the event Tech Conference the count of good pair is 7
```

---

4. Question No 4

Detecting Palindromic Permutations in Usernames (8 marks) You are part of the cybersecurity team for a social media platform. Your task is to ensure that usernames c permutations. A username is considered to have a palindromic permutation if you can rearrange its letters to form a palindrome. A palindrome is a string that reads the given a username, determine if any permutation of the username can form a palindrome.

Analysis:

- We need to the get the userID and userName from the user.
- We need to check the userName is Palindrome with its permutations.
- If it is means return true.
- Otherwise return false.

Code:

```
package assessment;

import java.util.*;

public class QuestionNo4 {
    public static boolean isPalindromicPermutation(String username) {
        HashMap<Character, Integer> map = new HashMap<>();
        for (char c : username.toCharArray()) {
            map.put(c, map.getOrDefault(c, 0) + 1);
        }

        int oddCount = 0;
        for (int count : map.values()) {
            if (count % 2 != 0) {
                oddCount++;
            }
        }

        return oddCount <= 1;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
```

```
        System.out.print("Enter the user id: ");
        String userId = sc.nextLine();
        System.out.print("Enter the username: ");
        String username = sc.nextLine();
        System.out.println(isPalindromicPermutation(username));
    }
}
```

Output:

```
Enter the user id: shabari2003
Enter the username: racecar
true
```

5.You have been tasked with developing an inventory management system for an online store.

The system should effectively store and manage product information, including the product's name, SKU (Stock Keeping Unit), quantity available, and price. To achiev solution using an ArrayList data structure to handle real-time inventory operations. Here's a detailed implementation of the system:

1. Start by defining a Product class that represents a product in the inventory. The class should have attributes such as the product's name, SKU, quantity, and price.
2. In the main inventory management class, create an ArrayList to store the products. This ArrayList will serve as the container for all product entries.
3. Implement a method to add a new product to the inventory. This method should take the necessary parameters (name, SKU, quantity, and price) and create a new the ArrayList.
4. Implement a method to remove a product from the inventory. This method should take the SKU of the product as a parameter. Iterate over the ArrayList and searc found, remove the product from the ArrayList.
5. Create a method to update the details of a product in the inventory. This method should take the SKU as a parameter and allow for modifying the product's attribu the ArrayList to find the product with the specified SKU and update its details accordingly.
6. Develop a method to search for a product in the inventory. This method should take a search query (SKU or name) as a parameter. Iterate over the ArrayList, comp each product. If a match is found, return the product object.
7. Implement a method to display a list of all products in the inventory. Iterate over the ArrayList and print the details of each product, including the name, SKU, qua

Analysis

- We need to create a Product class with the attributes name, SKU, quantity, and price.
- We need to create an ArrayList to store the products.
- We need to implement the following methods:
    1. Add a new product to the inventory.
    2. Remove a product from the inventory.
    3. Update the details of a product in the inventory.
    4. Search for a product in the inventory.
    5. Display a list of all products in the inventory.

Code:

```
package assessment;

import java.util.*;

class Product {
    String name;
    String SKU;
    int quantity;
    double price;

    public Product(String name, String SKU, int quantity, double price) {
```

```
        this.name = name;
        this.SKU = SKU;
        this.quantity = quantity;
        this.price = price;
    }
}

class InventoryManagement {
    ArrayList<Product> products = new ArrayList<>();

    public void addProduct(String name, String SKU, int quantity, double price) {
        Product product = new Product(name, SKU, quantity, price);
        products.add(product);
    }

    public void removeProduct(String SKU) {
        for (Product product : products) {
            if (product.SKU.equals(SKU)) {
                products.remove(product);
                break;
            }
        }
    }

    public void updateProduct(String SKU, int quantity, double price) {
        for (Product product : products) {
            if (product.SKU.equals(SKU)) {
                product.quantity = quantity;
                product.price = price;
                break;
            }
        }
    }

    public Product searchProduct(String query) {
        for (Product product : products) {
            if (product.SKU.equals(query) || product.name.equals(query)) {
                return product;
            }
        }
        return null;
    }

    public void displayProducts() {
        for (Product product : products) {
            System.out.println("Name: " + product.name);
            System.out.println("SKU: " + product.SKU);
            System.out.println("Quantity: " + product.quantity);
            System.out.println("Price: " + product.price);
            System.out.println();
        }
    }
}

public class QuestionNo5 {
    public static void main(String[] args) {
        InventoryManagement inventory = new InventoryManagement();
        System.out.println("Adding the below Product to the Inventory.");
        inventory.addProduct("Laptop", "Shabari123", 10, 500.0);
        inventory.addProduct("Mouse", "Sabari456", 20, 20.0);
        inventory.addProduct("Keyboard", "Shabari789", 15, 30.0);
        inventory.displayProducts();

        System.out.println("Removing Shabari456 from the inventory");
        inventory.removeProduct("Shabari456");
```

```
        System.out.println("Displaying All the product in the inventory");
        inventory.displayProducts();

        System.out.println("Updating Shabari123 in the Inventory.");
        inventory.updateProduct("Shabari123", 5, 600.0);

        System.out.println("Displaying All the product in the inventory");
        inventory.displayProducts();

        Product product = inventory.searchProduct("Shabari789");
        if (product != null) {
            System.out.println("Product found:");
            System.out.println("Name: " + product.name);
            System.out.println("SKU: " + product.SKU);
            System.out.println("Quantity: " + product.quantity);
            System.out.println("Price: " + product.price);
        } else {
            System.out.println("Product not found.");
        }
    }
}
```

Output

```
Adding the below Product to the Inventory.
Name: Laptop
SKU: Shabari123
Quantity: 10
Price: 500.0

Name: Mouse
SKU: Sabari456
Quantity: 20
Price: 20.0

Name: Keyboard
SKU: Shabari789
Quantity: 15
Price: 30.0

Removing Shabari456 from the inventory
Displaying All the product in the inventory
Name: Laptop
SKU: Shabari123
Quantity: 10
Price: 500.0

Name: Mouse
SKU: Sabari456
Quantity: 20
Price: 20.0

Name: Keyboard
SKU: Shabari789
Quantity: 15
Price: 30.0

Updating Shabari123 in the Inventory.
Displaying All the product in the inventory
Name: Laptop
SKU: Shabari123
Quantity: 5
Price: 600.0
```

```
Name: Mouse
SKU: Sabari456
Quantity: 20
Price: 20.0

Name: Keyboard
SKU: Shabari789
Quantity: 15
Price: 30.0

Product found:
Name: Keyboard
SKU: Shabari789
Quantity: 15
Price: 30.0
```