**1.  Create Account A and Account B with an initial balance of 5000 and 2500 respectively. Transfer amount of 1500 from Account A to B and an amount of 3000 from Account B to A.**

**Print the receipt with the following details after each transaction**

**Output:**

**Account id: 12344,**

**Name: XXXX,**

**Account Balance: Rs.\_\_\_\_**

**CODE :**

class BankAccount {

private int accountId;

private String name;

private double balance;

public BankAccount(int accountId, String name, double balance) {

this.accountId = accountId;

this.name = name;

this.balance = balance;

}

public void deposit(double amount) {

balance += amount;

}

public void withdraw(double amount) {

if (balance >= amount) {

balance -= amount;

} else {

System.out.println("Insufficient funds");

}

}

public void printReceipt() {

System.out.println("Account id: " + accountId);

System.out.println("Name: " + name);

System.out.println("Account Balance: Rs." + balance);

System.out.println("-----------------------");

}

}

public class Main {

public static void main(String[] args) {

BankAccount accountA = new BankAccount(12344, "Account A", 5000);

BankAccount accountB = new BankAccount(56789, "Account B", 2500);

accountA.withdraw(1500);

accountB.deposit(1500);

System.out.println("After transferring 1500 from Account A to Account B:");

accountA.printReceipt();

accountB.printReceipt();

accountB.withdraw(3000);

accountA.deposit(3000);

System.out.println("After transferring 3000 from Account B to Account A:");

accountA.printReceipt();

accountB.printReceipt();

}

}

**2.  Given an array and a partition size, you have to partition the array with that value , then we will specify the partition order, you have to merge based on that order**

**Input:**

**Array : 1 2 3 4 5**

**Partition size 2 (so the array will be partitioned as 1 2, 3 4, 5)**

**Partition order 3 2 1**

**Output:**

**5 3 4 1 2**

**CODE :**

import java.util.\*;

class ArrayPartition {

private int[] array;

private int partitionSize;

public ArrayPartition(int[] array, int partitionSize) {

this.array = array;

this.partitionSize = partitionSize;

}

public void partitionArray() {

ArrayList<int[]> partitions = new ArrayList<>();

for (int i = 0; i < array.length; i += partitionSize) {

int[] partition = Arrays.copyOfRange(array, i, Math.min(i + partitionSize, array.length));

partitions.add(partition);

}

for (int i = 0; i < partitions.size(); i++) {

array = partitions.get(i);

reverseArray(array);

partitions.set(i, array);

}

Collections.reverse(partitions);

mergeArray(partitions);

}

private void reverseArray(int[] arr) {

int start = 0;

int end = arr.length - 1;

while (start < end) {

int temp = arr[start];

arr[start] = arr[end];

arr[end] = temp;

start++;

end--;

}

}

private void mergeArray(ArrayList<int[]> partitions) {

ArrayList<Integer> merged = new ArrayList<>();

for (int i = 0; i < partitions.size(); i++) {

for (int j = 0; j < partitions.get(i).length; j++) {

merged.add(partitions.get(i)[j]);

}

}

for (int i = 0; i < merged.size(); i++) {

System.out.print(merged.get(i) + " ");

}

}

}

public class Main {

public static void main(String[] args) {

int[] inputArray = {1, 2, 3, 4, 5};

int partitionSize = 2;

int[] partitionOrder = {3, 2, 1};

ArrayPartition arrayPartition = new ArrayPartition(inputArray, partitionSize);

arrayPartition.partitionArray();

}

}

**3. A palindrome number - number that remains the same after reversing each digit of that number. A prime number - number that is divisible by only one or itself. A number that satisfies both the properties is said to be PalPrime Number.**

**Create a class PalPrime with a parameterised constructor PalPrime(int number, String message).**

**Given an positive integer array of numbers, you have to traverse the array and print the message “Number \_\_\_ is Prime/Palindrome/PalPrime”.**

**Note: Message should be printed via constructor of PalPrime class.**

**Input :**

**Array: [1, 34543, 565, 727, 10099]**

**Output -> Predict the output**

**CODE :**

import java.util.\*;

class PalPrime {

private int number;

private String message;

public PalPrime(int number, String message) {

this.number = number;

this.message = message;

System.out.println("Number " + number + " is " + message);

}

private boolean isPalindrome(int num) {

int original = num;

int reversed = 0;

while (num > 0) {

int remainder = num % 10;

reversed = reversed \* 10 + remainder;

num /= 10;

}

return original == reversed;

}

private boolean isPrime(int num) {

if (num <= 1) {

return false;

}

for (int i = 2; i <= Math.sqrt(num); i++) {

if (num % i == 0) {

return false;

}

}

return true;

}

public void checkPalPrime() {

boolean palindrome = isPalindrome(number);

boolean prime = isPrime(number);

if (palindrome && prime) {

message = "PalPrime";

} else if (palindrome) {

message = "Palindrome";

} else if (prime) {

message = "Prime";

} else {

message = "Neither Palindrome nor Prime";

}

System.out.println("Number " + number + " is " + message);

}

}

public class Main {

public static void main(String[] args) {

int[] numbers = {1, 34543, 565, 727, 10099};

for (int num : numbers) {

PalPrime palPrime = new PalPrime(num, "");

palPrime.checkPalPrime();

}

}

}