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.\_\_\_\_

public class Account {

private int accountId;

private String name;

private double balance;

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

this.accountId = accountId;

this.name = name;

this.balance = balance;

}

public void transfer(Account toAccount, double amount) {

if (balance >= amount) {

balance -= amount;

toAccount.balance += amount;

System.out.println("Transfer successful.");

} else {

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

}

}

public void printReceipt() {

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

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

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

}

public static void main(String[] args) {

Account accountA = new Account(12345, "John Doe", 5000);

Account accountB = new Account(54321, "Jane Doe", 2500);

// Transfer 1500 from A to B

accountA.transfer(accountB, 1500);

// Print receipts for both accounts

System.out.println("After first transaction:");

System.out.println("Account A:");

accountA.printReceipt();

System.out.println("Account B:");

accountB.printReceipt();

// Transfer 3000 from B to A

accountB.transfer(accountA, 3000);

// Print receipts for both accounts again

System.out.println("\nAfter second transaction:");

System.out.println("Account A:");

accountA.printReceipt();

System.out.println("Account B:");

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

public class ArrayPartitionMerge {

public static int[] partitionAndMerge(int[] arr, int partitionSize, int[] order) {

// Check for invalid partition size

if (partitionSize <= 0 || partitionSize > arr.length) {

throw new IllegalArgumentException("Invalid partition size");

}

// Create partitions

List<List<Integer>> partitions = new ArrayList<>();

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

partitions.add(Arrays.stream(arr, i, Math.min(i + partitionSize, arr.length)).boxed().collect(Collectors.toList()));

}

// Merge partitions based on order

int[] mergedArr = new int[arr.length];

int index = 0;

for (int i : order) {

for (int num : partitions.get(i - 1)) {

mergedArr[index++] = num;

}

}

return mergedArr;

}

public static void main(String[] args) {

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

int partitionSize = 2;

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

int[] mergedArr = partitionAndMerge(arr, partitionSize, order);

System.out.println("Merged array: " + Arrays.toString(mergedArr));

}

}

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

class PalPrime {

int number;

String message;

public PalPrime(int number, String message) {

this.number = number;

this.message = message;

// Check for palindrome and prime properties, and print the appropriate message

if (isPalindrome() && isPrime()) {

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

} else if (isPalindrome()) {

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

} else if (isPrime()) {

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

} else {

System.out.println(message + " " + number + " is neither Prime nor Palindrome");

}

}

boolean isPalindrome() {

int original = number;

int reversed = 0;

while (original > 0) {

int digit = original % 10;

reversed = reversed \* 10 + digit;

original /= 10;

}

return number == reversed;

}

boolean isPrime() {

if (number <= 1) {

return false;

}

for (int i = 2; i \* i <= number; i++) {

if (number % i == 0) {

return false;

}

}

return true;

}

}

public class Main {

public static void main(String[] args) {

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

String message = "Number";

for (int num : arr) {

new PalPrime(num, message);

}

}

}

**Output:**

Number 1 is Palindrome

Number 34543 is PalPrime

Number 565 is Palindrome

Number 727 is PalPrime

Number 10099 is Prime