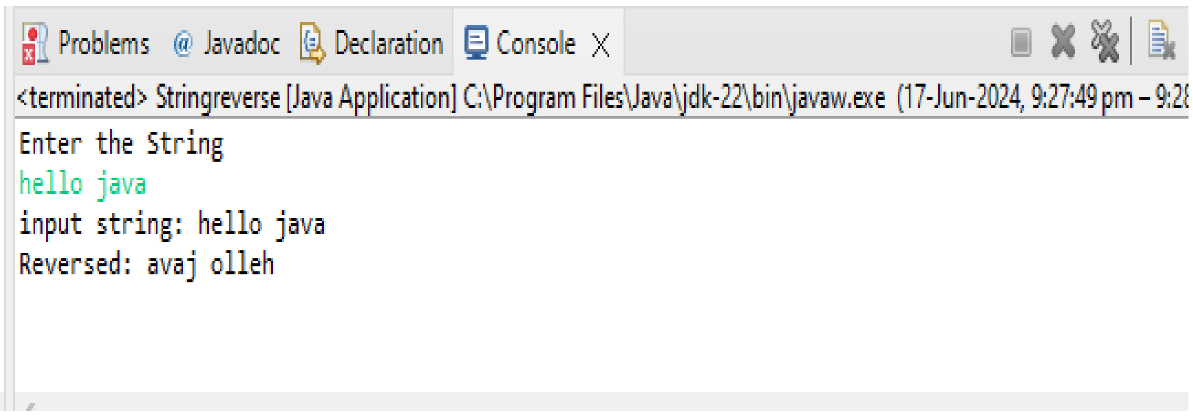
ASSIGNMENT-2

String API

1. Write a method to reverse a given string without using the built-in reverse method. The method should take a string as input and return the reversed string.

```
package assignment2;

import java.util.Scanner;

public class Stringreverse {
public static String reverseString(String str) {

char[] charArray = str.toCharArray();

int first = 0;
int last = charArray.length - 1;

while (first < last) {

char temp = charArray[first];
charArray[first] = charArray[last];
charArray[last] = temp;

first++;
last--;
}
return new String(charArray);
}

public static void main(String[] args) {
Scanner myobj = new Scanner(System.in);
System.out.println("Enter the String");
String input = myobj.nextLine();
String reversed = reverseString(input);
System.out.println("input string: " + input);
System.out.println("reversed: " + reversed);
myobj.close();
}
}
```

output



```
<terminated> Stringreverse [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (17-Jun-2024, 9:27:49 pm – 9:2
Enter the String
hello java
input string: hello java
Reversed: avaj olleh
```

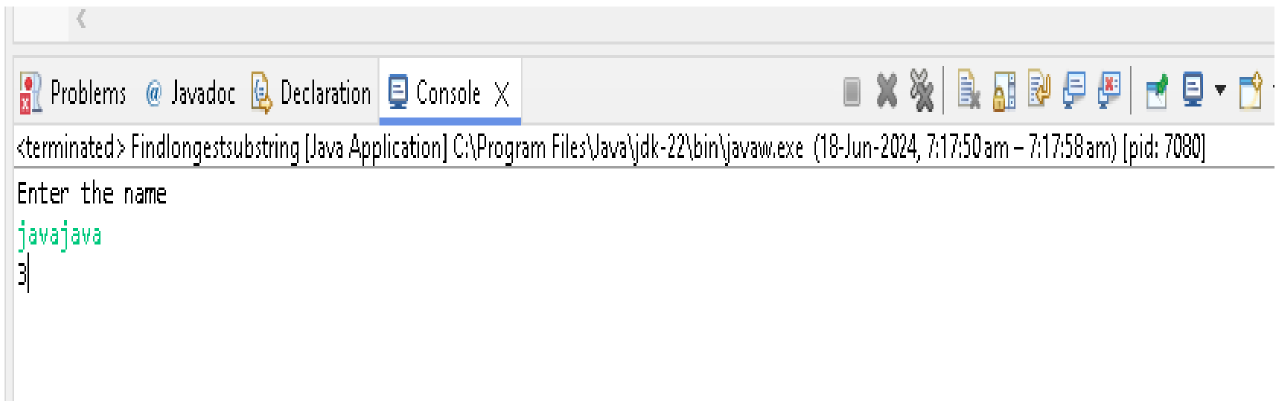
2. Write a method to find the length of the longest substring without repeating characters in a given string.

```
package assignment2;

import java.util.*;

public class Findlongestsubstring{
public static int lengthOfLongestSubstring(String s) {
```

```java
if (s == null || s.length() == 0) {
return 0;
}

int n = s.length();
int maxLength = 0;
int start = 0;
Map<Character, Integer> charIndexMap = new HashMap<>();

for (int end = 0; end < n; end++) {
char currentChar = s.charAt(end);
if (charIndexMap.containsKey(currentChar)) {
start = Math.max(charIndexMap.get(currentChar) + 1, start);
}
charIndexMap.put(currentChar, end);
maxLength = Math.max(maxLength, end - start + 1);
}

return maxLength;
}




public static void main(String[] args) {

System.out.println("Enter the name");
Scanner myobj = new Scanner(System.in);
String input = myobj.nextLine();

System.out.println(lengthOfLongestSubstring(input));
```

```
myobj.close();
}
}
```

Output


Problems Javadoc Declaration Console

```
<terminated> Findlongestsubstring [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (18-Jun-2024, 7:17:50 am – 7:17:58 am) [pid: 7080]
Enter the name
javajava
3
```

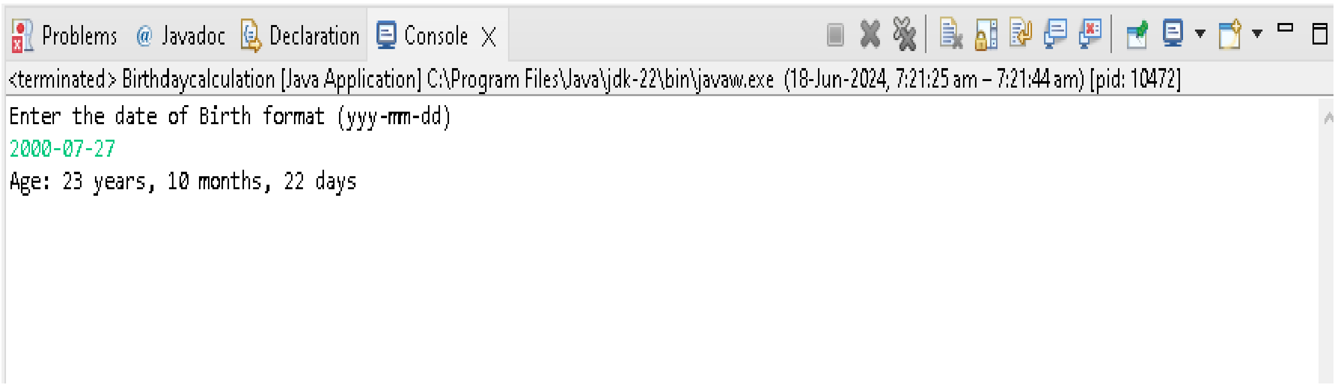
Date and Time API

3. Write a method that takes a person's birthdate in the format yyyy-MM-dd and calculates their age in years, months, and days. Use the java.time package to perform the calculations

```
package dateandtime;


import java.time.LocalDate;
import java.time.Period;
import java.time.format.DateTimeFormatter;
import java.util.Scanner;
```

```
public class Birthdaycalculation {

public static void main(String[] args) {
System.out.println("Enter the date of Birth format (yyy-
mm-dd)");
Scanner myobj=new Scanner(System.in);
String input=myobj.nextLine();
String birthdateStr = "2000-07-27";

String ageStr = calculateAge(input);
System.out.println("Age: " + ageStr);
myobj.close();
}

public static String calculateAge(String birthdateStr) {
LocalDate birthdate = LocalDate.parse(birthdateStr,
DateTimeFormatter.ofPattern("yyyy-MM-dd"));

LocalDate currentDate = LocalDate.now();

Period period = Period.between(birthdate, currentDate);
int years = period.getYears();
int months = period.getMonths();
int days = period.getDays();

return years + " years, " + months + " months, " + days + " days";
}
}
```

OUTPUT



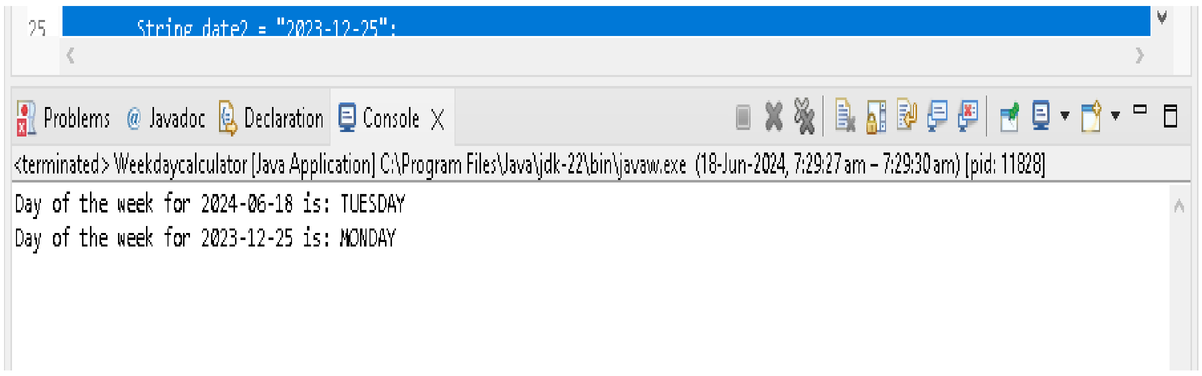
```
<terminated> Birthdaycalculation [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (18-Jun-2024, 7:21:25 am – 7:21:44 am) [pid: 10472]
Enter the date of Birth format (yyy-mm-dd)
2000-07-27
Age: 23 years, 10 months, 22 days
```

4. Write a method that takes a date in the format yyyy-MM-dd and returns the day of the week for that date.

```
package dateandtime;

import java.time.DayOfWeek;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
```

```
public class Weekdaycalculator {

public static String getDayOfWeek(String dateString) {

LocalDate date = LocalDate.parse(dateString);


DayOfWeek dayOfWeek = date.getDayOfWeek();


return dayOfWeek.toString();
}

public static void main(String[] args) {
String date1 = "2024-06-18";
String dayOfWeek1 = getDayOfWeek(date1);
System.out.println("Day of the week for " + date1 + " is: " +
dayOfWeek1);

String date2 = "2023-12-25";
String dayOfWeek2 = getDayOfWeek(date2);
System.out.println("Day of the week for " + date2 + " is: " +
dayOfWeek2);
}
}
```

OUTPUT



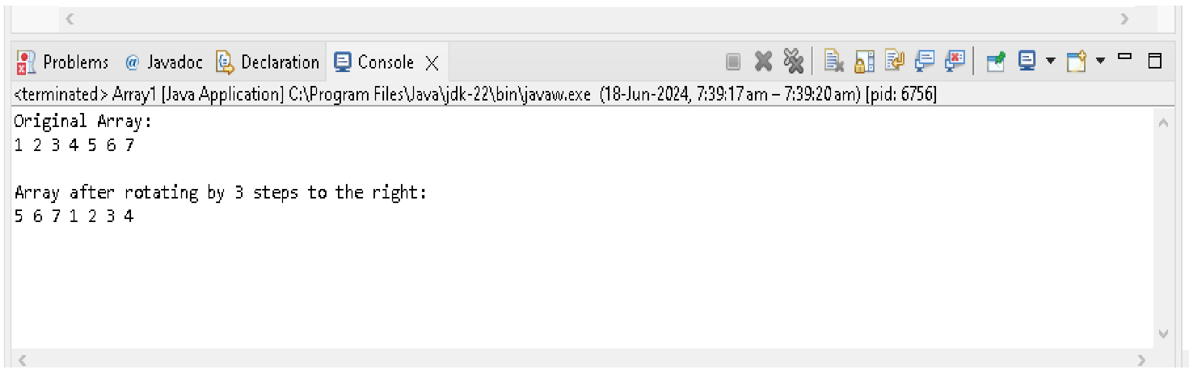
```
String date2 = "2023-12-25";
```

```
<terminated> Weekdaycalculator [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (18-Jun-2024, 7:29:27 am – 7:29:30 am) [pid: 11828]
Day of the week for 2024-06-18 is: TUESDAY
Day of the week for 2023-12-25 is: MONDAY
```

5. Write a method to rotate an array of integers by k steps to the right. For example, with k = 3, an array [1, 2, 3, 4, 5, 6, 7] becomes [5, 6, 7, 1, 2, 3, 4]. You must do this in-place with O(1) extra space.

```java
package arrray;

public class Array1 {

public static void rotate(int[] nums, int k) {
int n = nums.length;
k = k % n;

reverse(nums, 0, n - 1);

reverse(nums, 0, k - 1);

reverse(nums, k, n - 1);
}

private static void reverse(int[] nums, int start, int end) {
while (start < end) {
int temp = nums[start];
nums[start] = nums[end];
nums[end] = temp;
start++;
end--;
}
}

public static void main(String[] args) {
int[] nums = {1, 2, 3, 4, 5, 6, 7};
int k = 3;

System.out.println("Original Array:");
printArray(nums);

rotate(nums, k);

System.out.println("\nArray after rotating by " + k + " steps
to the right:");
printArray(nums);
}

private static void printArray(int[] arr) {
for (int num : arr) {
System.out.print(num + " ");
}
System.out.println();
}
}
```

OUTPUT

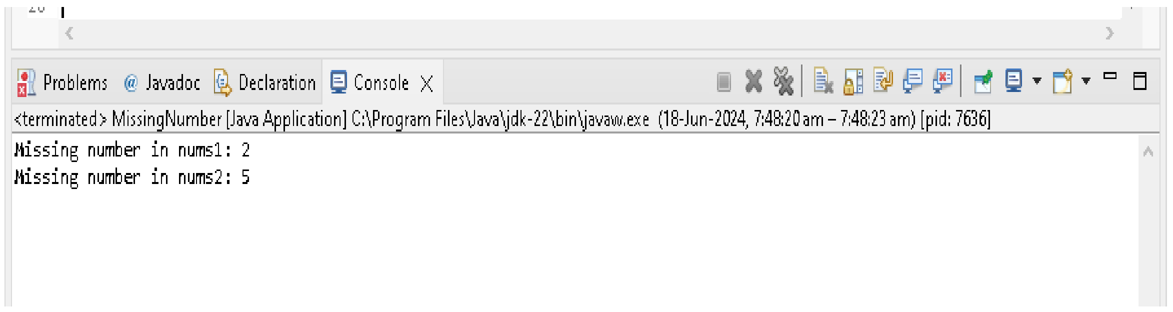

Problems @ Javadoc Declaration Console

```
<terminated> Array1 [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (18-Jun-2024, 7:39:17 am – 7:39:20 am) [pid: 6756]
Original Array:
1 2 3 4 5 6 7

Array after rotating by 3 steps to the right:
5 6 7 1 2 3 4
```

6. Given an array containing n distinct numbers taken from 0, 1, 2, ..., n, find the one that is missing from the array.

```
package arrray;

public class MissingNumber {

public static int findMissingNumber(int[] nums) {
int n = nums.length;
int expectedSum = n * (n + 1) / 2;

int actualSum = 0;
for (int num : nums) {
actualSum += num;
}

return expectedSum - actualSum;
}

public static void main(String[] args) {
int[] nums1 = {3, 0, 1};
System.out.println("Missing number in nums1: " +
findMissingNumber(nums1));
int[] nums2 = {9, 6, 4,1,2, 3, 7, 0,8};
System.out.println("Missing number in nums2: " +
findMissingNumber(nums2));
}
}
```

OUTPUT



```
<terminated> MissingNumber [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (18-Jun-2024, 7:48:20 am – 7:48:23 am) [pid: 7636]
Missing number in nums1: 2
Missing number in nums2: 5
```

Abstract class , inheritance , method overriding and polymorphism

7. Suppose you are designing a simple system for managing different types of vehicles. You want to create a hierarchy of vehicle classes to represent common behaviors and characteristics shared among them.

Create an abstract class called Vehicle with the following properties and methods:

Properties:

String make: representing the make (brand) of the vehicle.

String model: representing the model of the vehicle.

Abstract methods:

void accelerate(): representing the action of accelerating the vehicle.

void brake(): representing the action of applying brakes to the vehicle.

Create two subclasses of Vehicle:

Car: representing a car, with additional properties:

int numOfDoors: representing the number of doors in the car.

Implement the abstract methods accelerate() and brake() to print out appropriate messages for a car's acceleration and braking.

Motorcycle: representing a motorcycle, with additional properties:

boolean hasFairing: representing whether the motorcycle has a fairing or not (true/false).

Implement the abstract methods accelerate() and brake() to print out appropriate messages for a motorcycle's acceleration and braking.

Write a Main class with the main method to test your implementation. Instantiate objects of both Car and Motorcycle classes, and demonstrate their behaviors by calling the accelerate() and brake() methods.

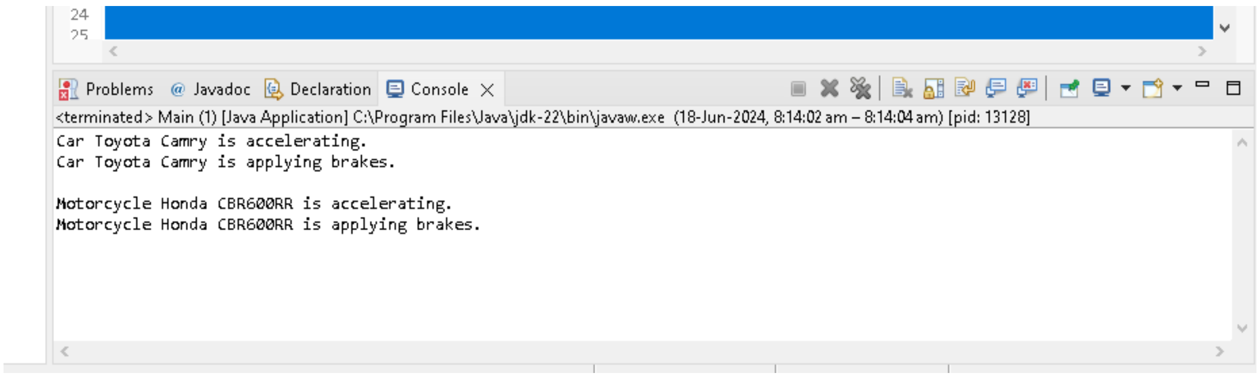
solution should demonstrate the use of abstract classes, inheritance, method overriding, and polymorphism in Java.

```java
package abstrac_inheritance_polimophism;

public class Main {
public static void main(String[] args) {
Car car = new Car("Toyota", "Camry", 4);
car.accelerate();
car.brake();

System.out.println();

Motorcycle motorcycle = new Motorcycle("Honda",
"CBR600RR", true);
motorcycle.accelerate();
motorcycle.brake();
}
}
```

```java
package abstrac_inheritance_polimophism;

abstract class Vehicle {
String make;
String model;

public Vehicle(String make, String model) {
this.make = make;
this.model = model;
}

abstract void accelerate();
abstract void brake();
}


package abstrac_inheritance_polimophism;

class Car extends Vehicle {
int numOfDoors;

public Car(String make, String model, int numOfDoors) {
super(make, model);
this.numOfDoors = numOfDoors;
}

@Override
void accelerate() {
System.out.println("Car " + make + " " + model + " is
accelerating.");
}

@Override
void brake() {
System.out.println("Car " + make + " " + model + " is applying
brakes.");
}
}

package abstrac_inheritance_polimophism;

class Motorcycle extends Vehicle {
boolean hasFairing;
```

```java
public Motorcycle(String make, String model, boolean
hasFairing) {
super(make, model);
this.hasFairing = hasFairing;
}

@Override
void accelerate() {
System.out.println("Motorcycle " + make + " " + model + " is
accelerating.");
}

@Override
void brake() {
System.out.println("Motorcycle " + make + " " + model + " is
applying brakes.");
}
}
```

OUTPUT



```
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe  (18-Jun-2024, 8:14:02 am – 8:14:04 am) [pid: 13128]
Car Toyota Camry is accelerating.
Car Toyota Camry is applying brakes.

Motorcycle Honda CBR600RR is accelerating.
Motorcycle Honda CBR600RR is applying brakes.
```

8. Let's design an online library system that manages various types of materials such as books,

magazines, and audiobooks. We'll create a hierarchy of classes to represent these materials,

utilizing abstract classes, inheritance, and polymorphism.

Abstract Class: LibraryMaterial

Properties:

String title: representing the title of the material.

String author: representing the author of the material.

Abstract methods:

void checkout(): representing the action of checking out the material.

void returnMaterial(): representing the action of returning the material.

Subclasses:

Book: Represents a book.

Additional properties:

int numberOfPages: representing the number of pages in the book.

Implement checkout() and returnMaterial() methods with appropriate messages.

Magazine: Represents a magazine.

Additional properties:

int issueNumber: representing the issue number of the magazine.

Implement checkout() and returnMaterial() methods with appropriate messages.

Audiobook: Represents an audiobook.

Additional properties:

String narrator: representing the narrator of the audiobook.

Implement checkout() and returnMaterial() methods with appropriate messages.

LibrarySystem Class:

Contains a list of available materials.

Methods to:

Add materials to the library.

Display available materials.

Allow users to checkout and return materials.

Main Class:

Contains the main method to test the system.

Demonstrates adding various materials to the library, checking out and returning them, and

displaying the available materials.

```
package librarymanagement;
```

```java
public class Main {
public static void main(String[] args) {
LibrarySystem library = new LibrarySystem();

Book book = new Book("thirukural", "thiruvalluvar", 400);
Magazine magazine = new Magazine("generalscience",
"Various Authors", 123);
Audiobook audiobook = new Audiobook("The Great Gatsby",
"F. Scott Fitzgerald", "Jake Gyllenhaal");
library.addMaterial(book);
library.addMaterial(magazine);
library.addMaterial(audiobook);

library.displayAvailableMaterials();

library.checkoutMaterial(book);
library.checkoutMaterial(magazine);
library.displayAvailableMaterials();

library.returnMaterial(book);
library.displayAvailableMaterials();
}
}


package librarymanagement;

abstract class LibraryMaterial {
String title;
String author;
boolean isCheckedOut;
public LibraryMaterial(String title, String author) {
this.title = title;
this.author = author;
this.isCheckedOut = false;
}

abstract void checkout();
abstract void returnMaterial();

public String getTitle() {
return title;
}

public String getAuthor() {
return author;
}

public boolean isCheckedOut() {
return isCheckedOut;
}

public void setCheckedOut(boolean checkedOut) {
isCheckedOut = checkedOut;
}
}
```

```java
package librarymanagement;
```

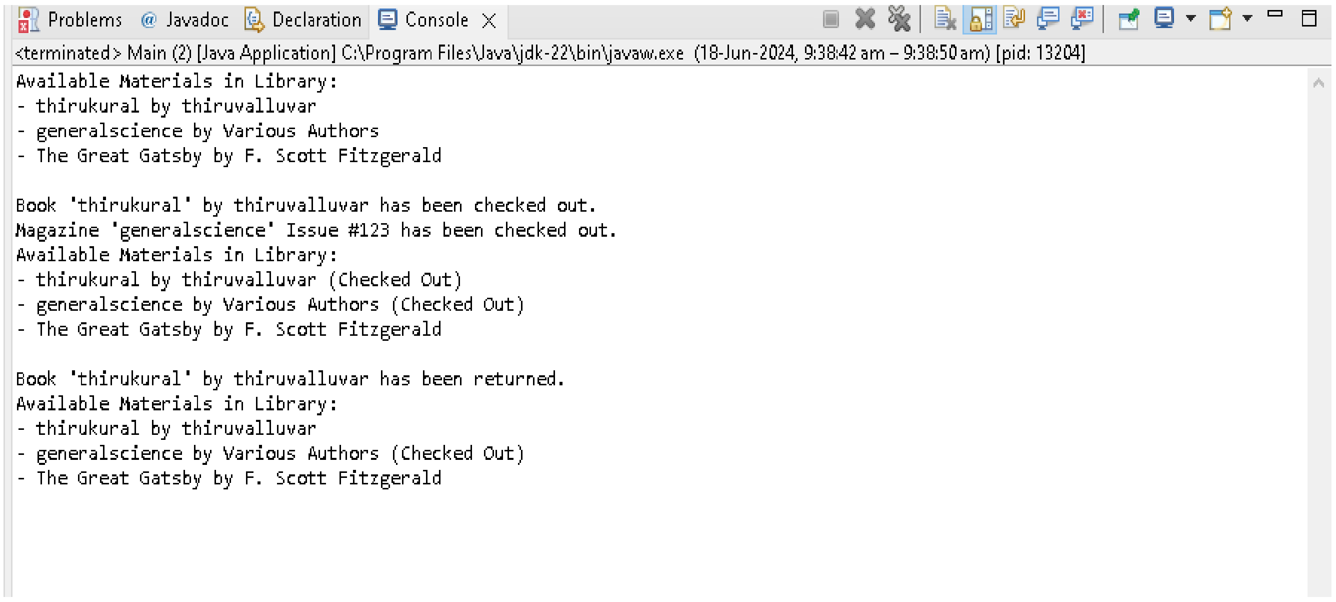
```
class Book extends LibraryMaterial {
int numberOfPages;

public Book(String title, String author, int numberOfPages) {
super(title, author);
this.numberOfPages = numberOfPages;
}

@Override
void checkout() {
System.out.println("Book '" + title + "' by " + author + " has
been checked out.");
setCheckedOut(true);
}

@Override
void returnMaterial() {
System.out.println("Book '" + title + "' by " + author + " has
been returned.");
setCheckedOut(false);
}
}



package librarymanagement;

class Magazine extends LibraryMaterial {
int issueNumber;

public Magazine(String title, String author, int issueNumber) {
```

```
super(title, author);
this.issueNumber = issueNumber;
}

@Override
void checkout() {
System.out.println("Magazine '" + title + "' Issue #" +
issueNumber + " has been checked out.");
setCheckedOut(true);
}

@Override
void returnMaterial() {
System.out.println("Magazine '" + title + "' Issue #" +
issueNumber + " has been returned.");
setCheckedOut(false);
}
}


package librarymanagement;

class Audiobook extends LibraryMaterial {
String narrator;

public Audiobook(String title, String author, String narrator) {
super(title, author);
this.narrator = narrator;
}

@Override
```

```java
void checkout() {
System.out.println("Audiobook '" + title + "' by " + author + "
narrated by " + narrator + " has been checked out.");
setCheckedOut(true);
}

@Override
void returnMaterial() {
System.out.println("Audiobook '" + title + "' by " + author + "
narrated by " + narrator + " has been returned.");
setCheckedOut(false);
}
}



package librarymanagement;

import java.util.ArrayList;
import java.util.List;

public class LibrarySystem {
private List<LibraryMaterial> availableMaterials;

public LibrarySystem() {
this.availableMaterials = new ArrayList<>();
}

public void addMaterial(LibraryMaterial material) {
availableMaterials.add(material);
}
```

```
public void displayAvailableMaterials() {
System.out.println("Available Materials in Library:");
for (LibraryMaterial material : availableMaterials) {
System.out.println("- " + material.getTitle() + " by " +
material.getAuthor() +
(material.isCheckedOut() ? " (Checked Out)" : ""));
}
System.out.println();
}

public void checkoutMaterial(LibraryMaterial material) {
if (!material.isCheckedOut()) {
material.checkout();
} else {
System.out.println("Sorry, " + material.getTitle() + " is already
checked out.");
}
}

public void returnMaterial(LibraryMaterial material) {
if (material.isCheckedOut()) {
material.returnMaterial();
} else {
System.out.println("This " + material.getTitle() + " was not
checked out.");
}
}
}
```

OUTPUT



```
<terminated> Main (2) [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe  (18-Jun-2024, 9:38:42 am – 9:38:50 am) [pid: 13204]
Available Materials in Library:
- thirukural by thiruvalluvar
- generalscience by Various Authors
- The Great Gatsby by F. Scott Fitzgerald

Book 'thirukural' by thiruvalluvar has been checked out.
Magazine 'generalscience' Issue #123 has been checked out.
Available Materials in Library:
- thirukural by thiruvalluvar (Checked Out)
- generalscience by Various Authors (Checked Out)
- The Great Gatsby by F. Scott Fitzgerald

Book 'thirukural' by thiruvalluvar has been returned.
Available Materials in Library:
- thirukural by thiruvalluvar
- generalscience by Various Authors (Checked Out)
- The Great Gatsby by F. Scott Fitzgerald
```

9. In a Human Resources (HR) application, there's a requirement to handle employee data,

including their personal information and salary details. Develop a solution to add new

employees to the system, which involves validating the input data and handling various

exceptions that may occur during the process.

Design a custom exception hierarchy to handle the following scenarios:

InvalidEmployeeDataException: This exception should be thrown when the provided employee

data is invalid or incomplete.

EmployeeAlreadyExistsException: This exception should be thrown when trying to add an employee with an ID that already exists in the system.

SalaryBelowMinimumException: This exception should be thrown when the provided salary is

below the minimum threshold set by the company.

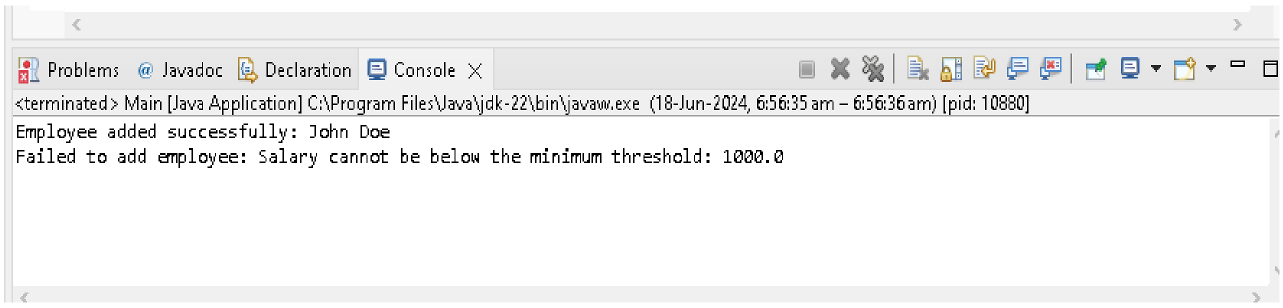
Write a method addEmployee() which takes employee ID, name, and salary as parameters. This

method should add a new employee to the system, handling the custom exceptions appropriately.

```
package exception;

import java.util.ArrayList;
import java.util.List;

class InvalidEmployeeDataException extends Exception {
public InvalidEmployeeDataException(String message) {
super(message);
}
}
class EmployeeAlreadyExistsException extends Exception {
public EmployeeAlreadyExistsException(String message) {
super(message);
}
}
class SalaryBelowMinimumException extends Exception {
public SalaryBelowMinimumException(String message) {
super(message);
}
```

```
}

// HRManager class
class HRManager {
private List<Employee> employeeList;
public HRManager() {
employeeList = new ArrayList<>();
}
public void addEmployee(String id, String name, double
salary)
throws
InvalidEmployeeDataException,EmployeeAlreadyExistsExcept
ion,
SalaryBelowMinimumException {
// Validate employee data
if (id == null || name == null || id.isEmpty() ||
name.isEmpty() || salary <= 0) {
throw new InvalidEmployeeDataException("Invalid employee
data provided");
}
// Check if employee already exists
for (Employee employee : employeeList) {
if (employee.getId().equals(id)) {
throw new EmployeeAlreadyExistsException("Employee with
ID " + id + " already exists");
}
}
// Check salary threshold
if (salary < Employee.getMinimumSalary()) {
throw new SalaryBelowMinimumException("Salary cannot
be below the minimum threshold: " +
Employee.getMinimumSalary());
}
// Add employee to the list
employeeList.add(new Employee(id, name, salary));
System.out.println("Employee added successfully: " + name);
}
}
package exception;

//Main class
public class Main {
public static void main(String[] args) {
HRManager hrManager = new HRManager();
// Test adding employees
try {
hrManager.addEmployee("1001", "John Doe", 1500);
hrManager.addEmployee("1002", "Jane Smith", 800);
} catch (InvalidEmployeeDataException |
EmployeeAlreadyExistsException |
SalaryBelowMinimumException e) {
System.out.println("Failed to add employee: " +
e.getMessage());
}
}
}
package exception;
//Employee class
public class Employee {
private String id;
private String name;
private double salary;
private static final double MINIMUM_SALARY = 1000; // Example minimum salary threshold
public Employee(String id, String name, double salary) {
this.id = id;
this.name = name;
this.salary = salary;
}
public String getId() {
return id;
}
public String getName() {
return name;
}
public double getSalary() {
return salary;
}
public static double getMinimumSalary() {
return MINIMUM_SALARY;
}
}
```

OUTPUT



```
<terminated> Main [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (18-Jun-2024, 6:56:35 am – 6:56:36 am) [pid: 10880]
Employee added successfully: John Doe
Failed to add employee: Salary cannot be below the minimum threshold: 1000.0
```

File Handling

10. You are developing a file management system for a small company. The system needs to

handle employee records stored in a file. Each employee record consists of the following

information:

Employee ID (unique identifier)

Name

Department

Salary

Design a Java program to manage these employee records using file handling. Your program should

provide the following functionalities:

Add a new employee record to the file.

Display all employee records from the file.

Search for an employee record by ID and display the details.

Update the salary of an employee by ID.

Delete an employee record by ID.

Ensure proper exception handling and validation throughout your program.

```
package filehandling;
import java.io.*;
import java.util.Scanner;
class Employee {
private String id;
private String name;
private String department;
private double salary;
public Employee(String id, String name, String department,
double salary) {
this.id = id;
this.name = name;
this.department = department;
this.salary = salary;
}
}


package filehandling;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;

//Generte Getter setter
//Generate toString
public class FileEmployeeManagement {
private static final String FILE_NAME =
"employee_records.txt";
```

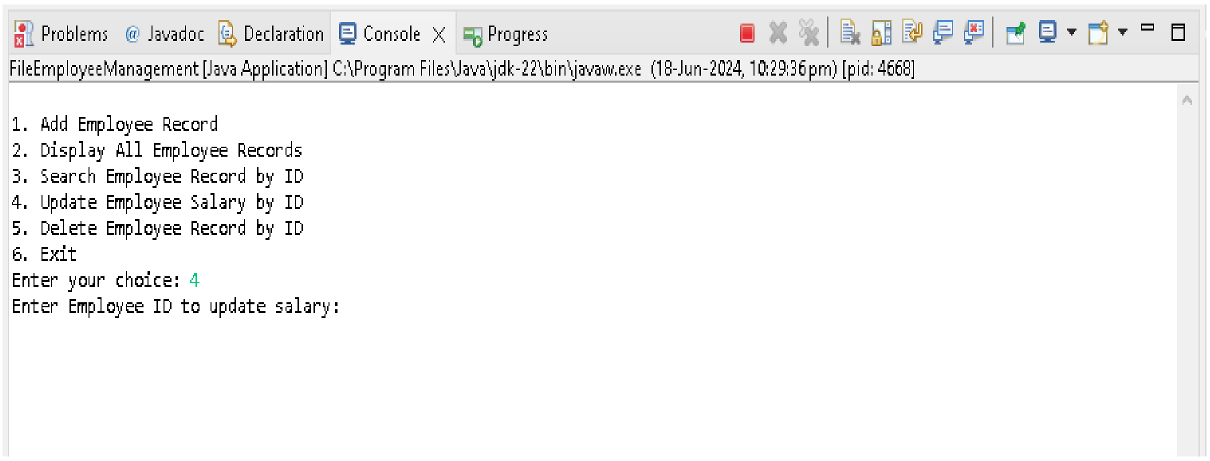
```
public static void main(String[] args) {
try {
File file = new File(FILE_NAME);
if (!file.exists()) {
file.createNewFile();
}
Scanner scanner = new Scanner(System.in);
int choice;
do {
System.out.println("\n1. Add Employee Record");
System.out.println("2. Display All Employee Records");
System.out.println("3. Search Employee Record by ID");
System.out.println("4. Update Employee Salary by ID");
System.out.println("5. Delete Employee Record by ID");
System.out.println("6. Exit");
System.out.print("Enter your choice: ");
choice = scanner.nextInt();
scanner.nextLine(); // Consume newline
switch (choice) {
case 1:
addEmployeeRecord();
break;
case 2:
displayAllEmployeeRecords();
break;
case 3:
searchEmployeeByID();
break;
case 4:
updateEmployeeSalary();
break;
case 5:
deleteEmployeeRecord();
break;
case 6:
System.out.println("Exiting...");
break;
default:
System.out.println("Invalid choice. Please enter a number
between 1 and 6.");
}
} while (choice != 6);
} catch (IOException e) {
System.out.println("Error: " + e.getMessage());
}
}
private static void addEmployeeRecord() throws IOException
{
//Get Employee data from user
// Create employee object say employee and pass it to
employe constructor
try (BufferedWriter writer = new BufferedWriter(new
FileWriter(FILE_NAME, true))) {
writer.write(employee.toString());
writer.newLine();
System.out.println("Employee record added successfully.");
}
}
private static void displayAllEmployeeRecords() throws
IOException {
try (BufferedReader reader = new BufferedReader(new
FileReader(FILE_NAME))) {
```

```
String line;
System.out.println("\nEmployee Records:");
while ((line = reader.readLine()) != null) {
System.out.println(line);
}
}
}
private static void searchEmployeeByID() throws
IOException {
//Get the employee id from user to be search in a variable
called searchID
try (BufferedReader reader = new BufferedReader(new
FileReader(FILE_NAME))) {
String line;
boolean found = false;
while ((line = reader.readLine()) != null) {
if (line.startsWith("ID: " + searchID)) {
System.out.println("Employee Record found:");
System.out.println(line);
found = true;
break;
}
}
if (!found) {
System.out.println("Employee with ID " + searchID + " not
found.");
}
}
}
private static void updateEmployeeSalary() throws
IOException {
```

```
Scanner scanner = new Scanner(System.in);
System.out.print("Enter Employee ID to update salary: ");
String updateID = scanner.nextLine();
StringBuilder newRecords = new StringBuilder();
boolean updated = false;
try (BufferedReader reader = new BufferedReader(new
FileReader(FILE_NAME))) {
String line;
while ((line = reader.readLine()) != null) {
if (line.startsWith("ID: " + updateID)) {
String[] parts = line.split(", ");
System.out.print("Enter new Salary for " + parts[1] + ": $");
double newSalary = scanner.nextDouble();
parts[3] = "Salary: $" + newSalary;
line = String.join(", ", parts);
updated = true;
}
newRecords.append(line).append("\n");
}
}
if (!updated) {
System.out.println("Employee with ID " + updateID + " not
found.");
} else {
try (BufferedWriter writer = new BufferedWriter(new
FileWriter(FILE_NAME))) {
writer.write(newRecords.toString());
System.out.println("Salary updated successfully.");
}
}
}
```

```
private static void deleteEmployeeRecord() throws
IOException {
//Get deleteID from user
StringBuilder newRecords = new StringBuilder();
boolean deleted = false;
try (BufferedReader reader = new BufferedReader(new
FileReader(FILE_NAME))) {
String line;
while ((line = reader.readLine()) != null) {
if (!line.startsWith("ID: " + deleteID)) {
newRecords.append(line).append("\n");
} else {
deleted = true;
}
}
}
if (!deleted) {
System.out.println("Employee with ID " + deleteID + " not
found.");
} else {
try (BufferedWriter writer = new BufferedWriter(new
FileWriter(FILE_NAME))) {
writer.write(newRecords.toString());
System.out.println("Employee record deleted successfully.");
}
}
}
}
```

OUTPUT



```
FileEmployeeManagement [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe  (18-Jun-2024, 10:29:36 pm) [pid: 4668]

1. Add Employee Record
2. Display All Employee Records
3. Search Employee Record by ID
4. Update Employee Salary by ID
5. Delete Employee Record by ID
6. Exit
Enter your choice: 4
Enter Employee ID to update salary:
```

Concurrency

11. Scenario: Imagine you're developing an online ticket booking system for a popular event

management company. This system needs to handle a large number of users trying to book

tickets simultaneously for various events. To ensure scalability and performance, you decide to

implement a concurrency mechanism.

Design a Java program for the ticket booking system with concurrency support. Your system should

include the following features:

Event Management:

Each event has a limited number of tickets available.

Users can book tickets for multiple events simultaneously.

Concurrency Control:

Implement a concurrency mechanism to handle multiple users trying to book tickets for the same

event simultaneously.

Ensure that tickets are not oversold or double-booked.

User Interaction:

Provide options for users to select events and book tickets.

Display available tickets for each event.

Inform users about the success or failure of their booking attempts.

Scalability:

Ensure that the system can handle a large number of simultaneous booking requests efficiently.

Minimize the chances of deadlock or race conditions.

Exception Handling:

Handle exceptions related to concurrency issues, such as overselling tickets or data

inconsistency.

Provide appropriate error messages and gracefully handle failures.

Design your solution considering the importance of thread safety, synchronization, and efficient

resource management in a concurrent environment.

```java
package concurency;

import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;
class Events {
private String eventName;
private int availableTickets;
private Lock lock;

public Event(String eventName, int availableTickets) {
this.eventName = eventName;
this.availableTickets = availableTickets;
this.lock = new ReentrantLock();
}
public boolean bookTickets(int numTickets) {
lock.lock();
try {
} finally {
lock.unlock();
}
}
}
```

```java
package concurency;

public class TicketBookingApp {
public static void main(String[] args) {
// Create array of no of event objects
```

```
TicketBookingSystem bookingSystem = new
TicketBookingSystem(events);
// Simulate multiple users trying to book tickets
simultaneously
Thread user1 = new Thread(() -> {
bookingSystem.bookTickets("Concert", 2);
});
Thread user2 = new Thread(() -> {
bookingSystem.bookTickets("Conference", 5);
});
Thread user3 = new Thread(() -> {
bookingSystem.bookTickets("Workshop", 3);
});
// Start each thread
// Display available tickets
bookingSystem.displayAvailableTickets();
}
}

package concurency;

import java.awt.~~Event~~;

class TicketBookingSystem {
private ~~Event~~[] events;
public TicketBookingSystem(~~Event~~[] events) {
this.events = events;
}
public void bookTickets(String eventName, int numTickets) {
for (~~Event~~ event : events) {
if (event.getEventName().equals(eventName)) {
```

```
if (event.bookTickets(numTickets)) {
System.out.println(numTickets + " tickets booked for event: "
+ eventName);
} else {
System.out.println("Failed to book tickets for event: " +
eventName + ". Insufficient tickets.");
}
return;
}
}
System.out.println("Event not found: " + eventName);
}
public void displayAvailableTickets() {
System.out.println("\nAvailable Tickets:");
// display the event and ticket details
}
}
```

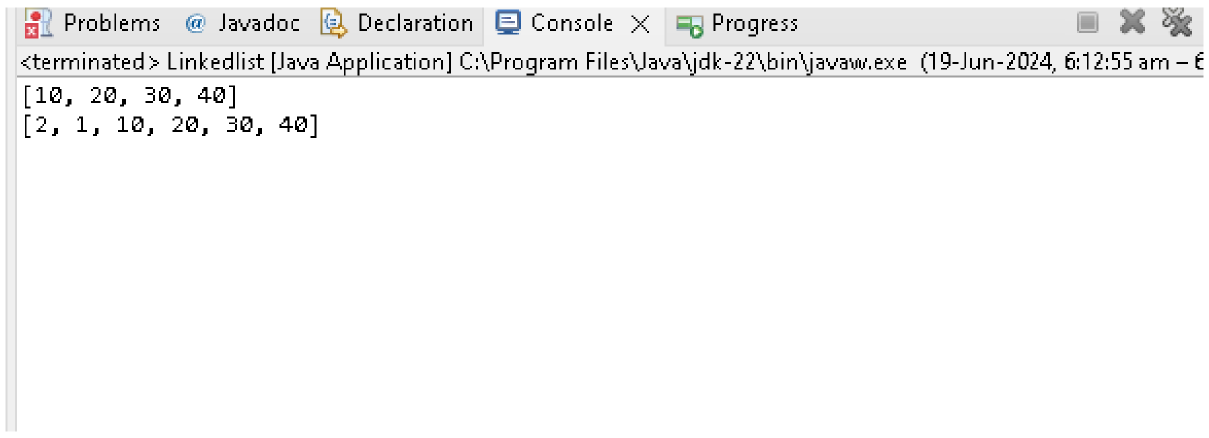
Collection

18. How do you insert an element at the head and tail of a LinkedList?

```
package collection;
```

```
import java.util.LinkedList;

public class Linkedlist
        {
        public static void main(String[] args)
        {
        LinkedList<Integer> list = new LinkedList<Integer>();
        //Adding elements at the end of the list
        list.add(10);
        list.addLast(20);
        list.offer(30);
        list.offerLast(40);
        //Printing the elements of list
        System.out.println(list); //Output : [10, 20, 30, 40]
        //Adding elements at the beginning of the list
        list.offerFirst(1);
        list.addFirst(2);
        //Printing the elements of list
        System.out.println(list); //Output : [2, 1, 10, 20, 30, 40]
        }
        }
```

OUTPUT

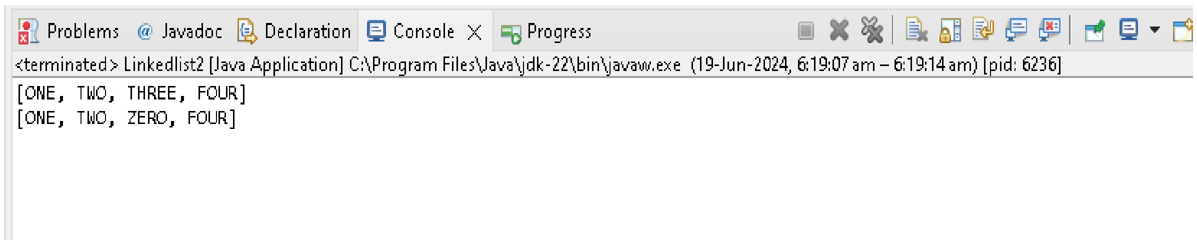


```
<terminated> Linkedlist [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (19-Jun-2024, 6:12:55 am – 6
[10, 20, 30, 40]
[2, 1, 10, 20, 30, 40]
```

19. How do you replace an element at a specific position of a LinkedList with the given element?

```
package collection;

import java.util.LinkedList;

public class Linkedlist2 {
	public static void main(String[] args)
	{
	LinkedList<String> list = new LinkedList<String>();
	list.add("ONE");
	list.add("TWO");
	list.add("THREE");
	list.add("FOUR");
	System.out.println(list);
	list.set(2, "ZERO");
	System.out.println(list);
	}
	}
```

OUTPUT



```
<terminated> Linkedlist2 [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe  (19-Jun-2024, 6:19:07 am – 6:19:14 am) [pid: 6236]
[ONE, TWO, THREE, FOUR]
[ONE, TWO, ZERO, FOUR]
```

20: Collection Implementation in a Task Management System

Introduction:

In a task management system, efficient handling of tasks and task lists is crucial for organizing and

managing projects effectively. Collections, a fundamental concept in Java, play a significant role in

storing

and manipulating task-related data structures. This case study explores the implementation of

collections

in a task management system and provides a solution along with test cases to ensure its

correctness.

Problem Statement:

A task management system needs to store and manage various tasks belonging to different

categories

such as personal, work, and errands. Each task has its unique attributes such as title, description,

due date, and priority. The system requires a flexible and scalable solution to store task, remove a task

on completion and sort the tasks

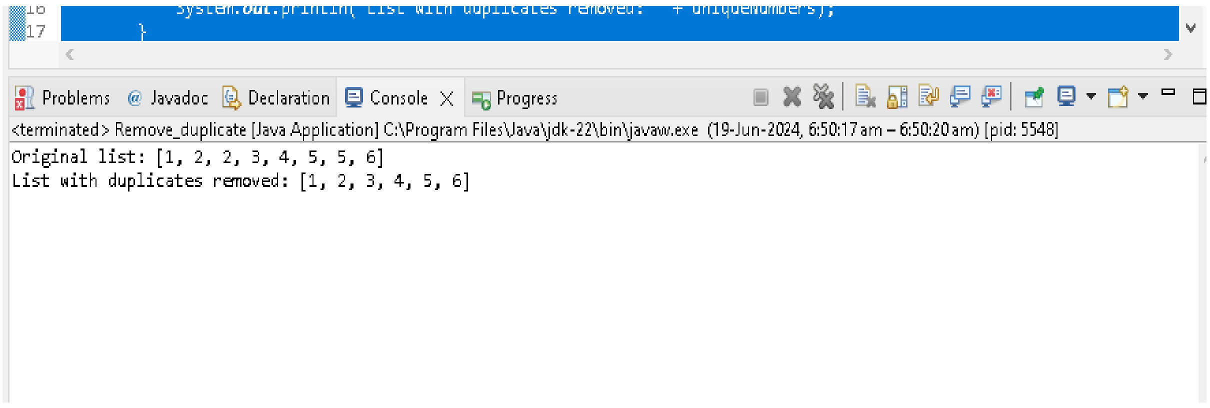
```
package collection;
        import java.time.LocalDate;
        public class Task {
         private String title;
         private String description;
         private LocalDate dueDate;
         private int priority;
         public Task(String title, String description, LocalDate
dueDate, int priority) {
         this.title = title;
         this.description = description;
         this.dueDate = dueDate;
         this.priority = priority;
         }
        }

package collection;

        import java.util.ArrayList;
        import java.util.ArrayList;
        import java.util.Collection;
        import java.util.Collections;
        import java.util.List;
        public class TaskList {
         private List<Task> tasks;
         public TaskList() {
         tasks = new ArrayList<Task>();
         }
         public void addTask(Task task) {
         tasks.add(task);
         }
         public void removeTask(Task task) {
         tasks.remove(task);
         }
         public List<Task> getTasks() {
         return tasks;
         }
         public void sortTasksByName() {
         Collections.sort(tasks, new TaskNameComparator());
         }
         }

package collection;
        import java.util.Comparator;
        public class TaskNameComparator implements
Comparator<Task> {
         public int compare(Task task1, Task task2) {
         return task1.getTitle().compareTo(task2.getTitle());
         }
        }

package collection;

        import java.time.LocalDate;
        public class Main {
         public static void main(String[] args) {
         // Create tasks
         Task task1 = new Task("Complete assignment", "Finish
the programming assignment\"",
        LocalDate.now().plusDays(5), 1);
         Task task2 = new Task("Buy groceries", "Purchase fruits
and vegetables",
        LocalDate.now().plusDays(2), 2);
         Task task3 = new Task("Study Assignment", "Study and
Complete the givne assignment\" "
        , LocalDate.now().plusDays(2), 2);
         Task task4 = new Task("Clean the room", "Wipe and
arrange the items in room",
        LocalDate.now().plusDays(2), 2);



         }


}
```

JAVA8

21. How do you remove duplicate elements from a list using Java 8 streams

```
package java8;
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;
public class Remove_duplicate {


	public static void main(String[] args) {
	    List<Integer> numbers = Arrays.asList(1, 2, 2, 3, 4, 5,
5, 6);

	    List<Object> uniqueNumbers = numbers.stream()
	                                    .distinct()
	                                    .collect(Collectors.toList());

	    System.out.println("Original list: " + numbers);
	    System.out.println("List with duplicates removed: "
+ uniqueNumbers);
	}
    }
```

OUTPUT



```
        System.out.println("List with duplicates removed: " + uniqueNumbers);
    }
```

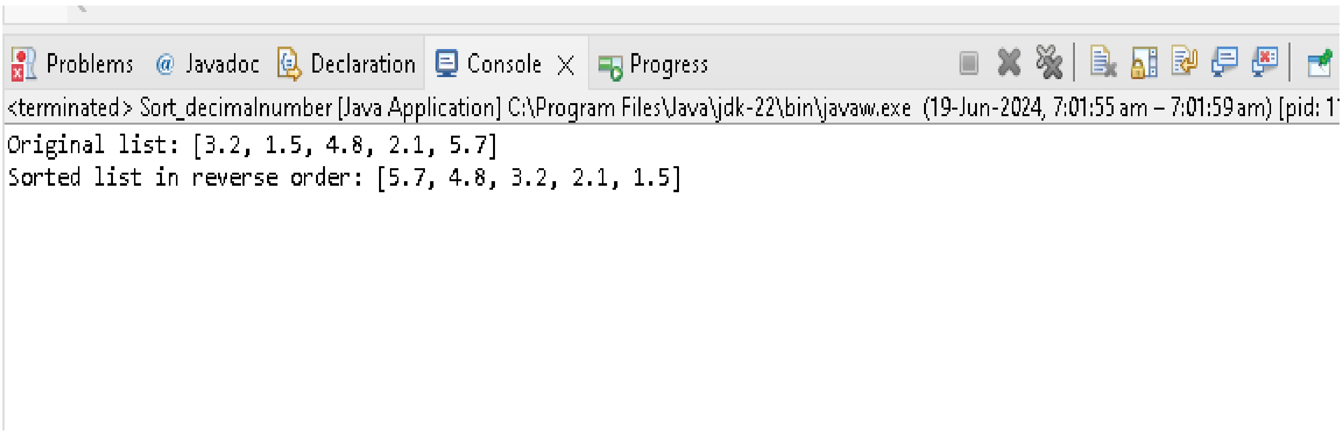
```
<terminated> Remove_duplicate [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe  (19-Jun-2024, 6:50:17 am – 6:50:20 am) [pid: 5548]
Original list: [1, 2, 2, 3, 4, 5, 5, 6]
List with duplicates removed: [1, 2, 3, 4, 5, 6]
```

22. How do you sort the given list of decimals in reverse order?

```
package java8;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;
import java.util.stream.Collectors;

public class Sort_decimalnumber {

	public static void main(String[] args) {
	    List<Double> decimals = Arrays.asList(3.2, 1.5, 4.8,
2.1, 5.7);

	    List<Object> sortedDecimals = decimals.stream()
	                                      .sorted()
	                                      .collect(Collectors.toList());

	    // Reverse the sorted list
	    Collections.reverse(sortedDecimals);

	    System.out.println("Original list: " + decimals);
```

```
		System.out.println("Sorted list in reverse order: " +
sortedDecimals);
	}
}
```

OUTPUT

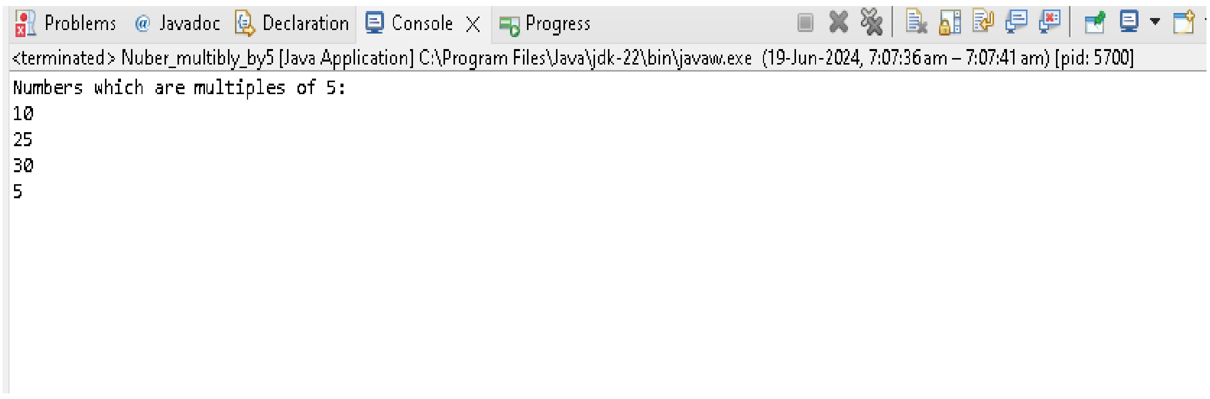


```
<terminated> Sort_decimalnumber [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe  (19-Jun-2024, 7:01:55 am – 7:01:59 am) [pid: 1
Original list: [3.2, 1.5, 4.8, 2.1, 5.7]
Sorted list in reverse order: [5.7, 4.8, 3.2, 2.1, 1.5]
```

23. From the given list of integers, print the numbers which are multiples of 5?

```
package java8;
import java.util.Arrays;
import java.util.List;

public class Nuber_multibly_by5 {
	public static void main(String[] args) {
	   List<Integer> numbers = Arrays.asList(1,2,9,10, 7, 
25, 14, 30, 18, 5);


	   System.out.println("Numbers which are multiples of 
5:");
	   numbers.stream()
	        .filter(num -> num % 5 == 0)
	        .forEach(System.out::println);
	}
    }
```

OUTPUT



```
<terminated> Nuber_multibly_by5 [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (19-Jun-2024, 7:07:36 am – 7:07:41 am) [pid: 5700]
Numbers which are multiples of 5:
10
25
30
5
```

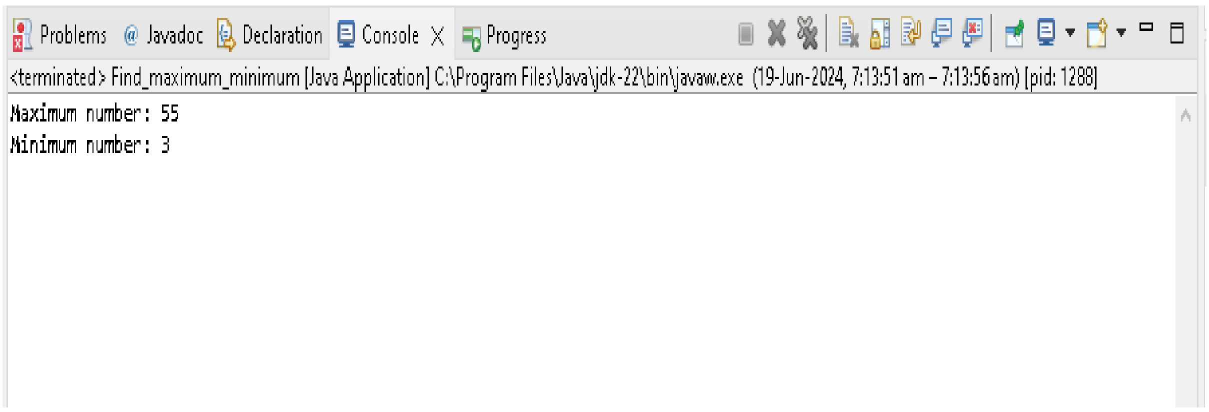
24. Given a list of integers, find maximum and minimum of those numbers

```
package java8;
import java.util.Arrays;
import java.util.List;
import java.util.Optional;
```

```
public class Find_maximum_minimum {

	public static void main(String[] args) {
	   List<Integer> numbers = Arrays.asList(3,10, 7, 25, 
14,55, 30, 18, 5);


	   Optional<Integer> max = numbers.stream()
	                          .max(Integer::compareTo);


	   Optional<Integer> min = numbers.stream()
	                          .min(Integer::compareTo);


	   if (max.isPresent()) {
	      System.out.println("Maximum number: " + 
max.get());
	   } else {
	      System.out.println("No maximum found (empty 
list)");
	   }

	   if (min.isPresent()) {
	      System.out.println("Minimum number: " + 
min.get());
	   } else {
	      System.out.println("No minimum found (empty 
list)");
```

```
            }
        }
    }
```

OUTPUT

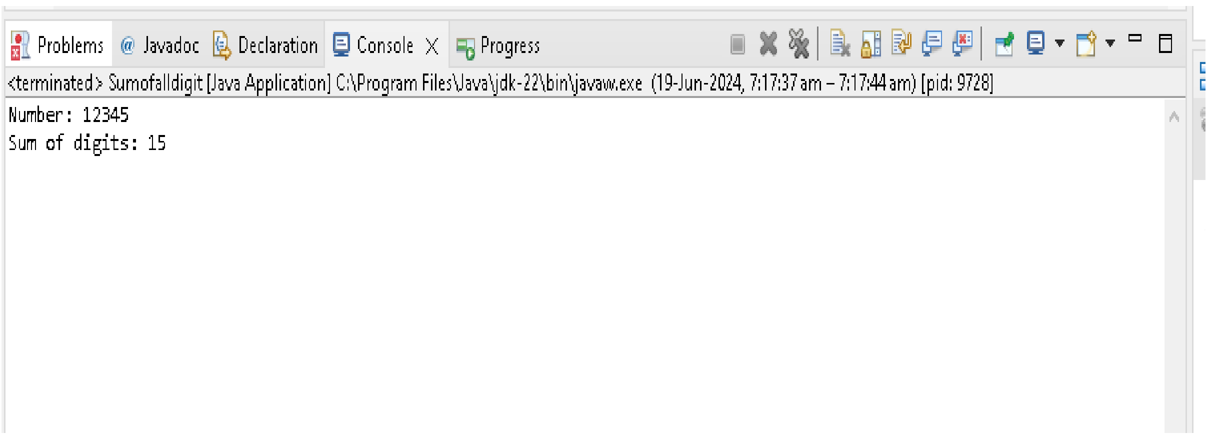


```
<terminated> Find_maximum_minimum [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (19-Jun-2024, 7:13:51 am – 7:13:56 am) [pid: 1288]
Maximum number: 55
Minimum number: 3
```

25. Find sum of all digits of a number in Java 8?

```
package java8;

public class Sumofalldigit {

        public static void main(String[] args) {
            int number = 12345;


            int sumOfDigits = String.valueOf(number)
                            .chars()
                            .map(Character::getNumericValue)
                            .sum();
            System.out.println("Number: " + number);
            System.out.println("Sum of digits: " + sumOfDigits);
        }
    }
```

OUTPUT

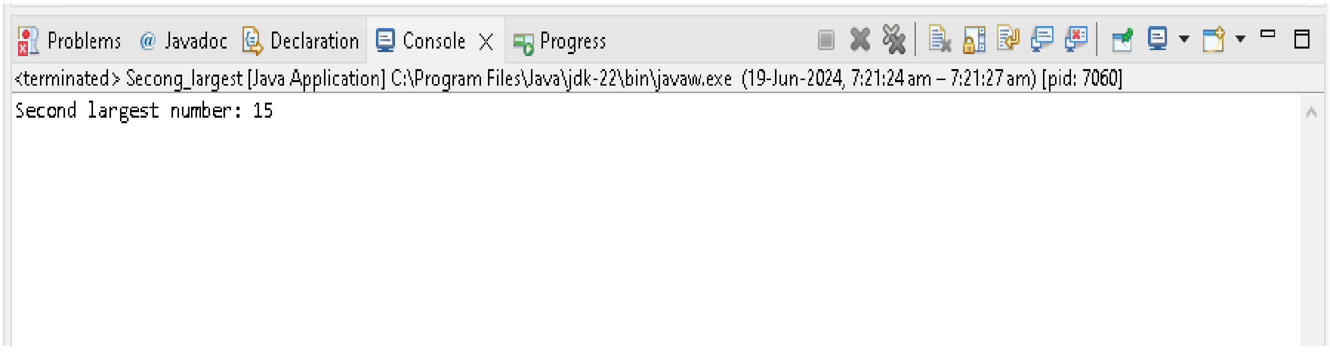


```
<terminated> Sumofalldigit [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (19-Jun-2024, 7:17:37 am – 7:17:44 am) [pid: 9728]
Number: 12345
Sum of digits: 15
```

26. Find second largest number in an integer array?

```
package java8;

public class Secong_largest {
    public static void main(String[] args) {
        int[] numbers = {10, 5, 8, 20, 3, 15};

        if (numbers.length < 2) {
            System.out.println("Array should have at least two elements");
            return;
        }
        int firstMax = Integer.MIN_VALUE;
        int secondMax = Integer.MIN_VALUE;

        for (int num : numbers) {
            if (num > firstMax) {
                secondMax = firstMax;
                firstMax = num;
            } else if (num > secondMax && num != firstMax) {
                secondMax = num;
            }
        }

        if (secondMax == Integer.MIN_VALUE) {
            System.out.println("There is no second largest
element");
        } else {
            System.out.println("Second largest number: " +
secondMax);
        }
    }
}
```

OUTPUT

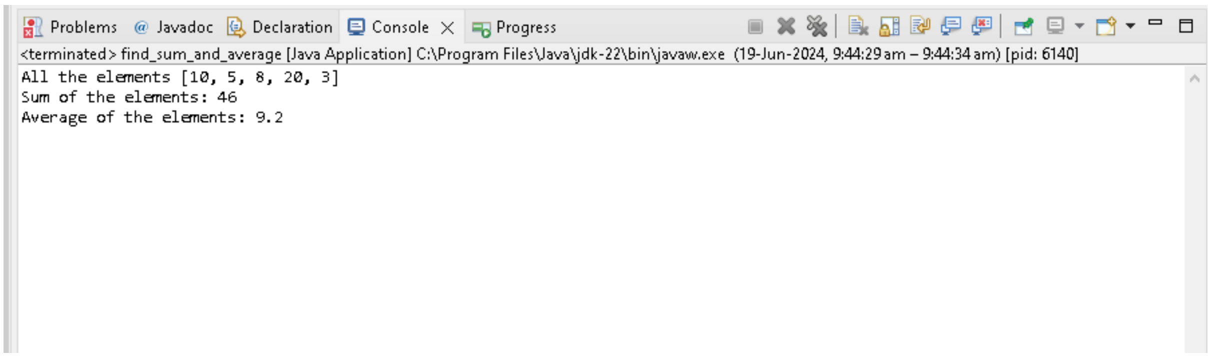


```
<terminated> Secong_largest [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (19-Jun-2024, 7:21:24 am – 7:21:27 am) [pid: 7060]
Second largest number: 15
```

29. Given an integer array, find sum and average of all elements

```
package java8;

public class find_sum_and_average {
	public static void main(String[] args) {
	   int[] numbers = {10, 5, 8, 20, 3};

	   int sum = 0;
	   for (int num : numbers) {
	      sum += num;
	   }

	   double average = (double) sum / numbers.length;

	   System.out.println("All the elements " +
java.util.Arrays.toString(numbers));
	   System.out.println("Sum of the elements: " + sum);
	   System.out.println("Average of the elements: " +
average);
	}
      }
```

OUTPUT



Problems @ Javadoc Declaration Console Progress

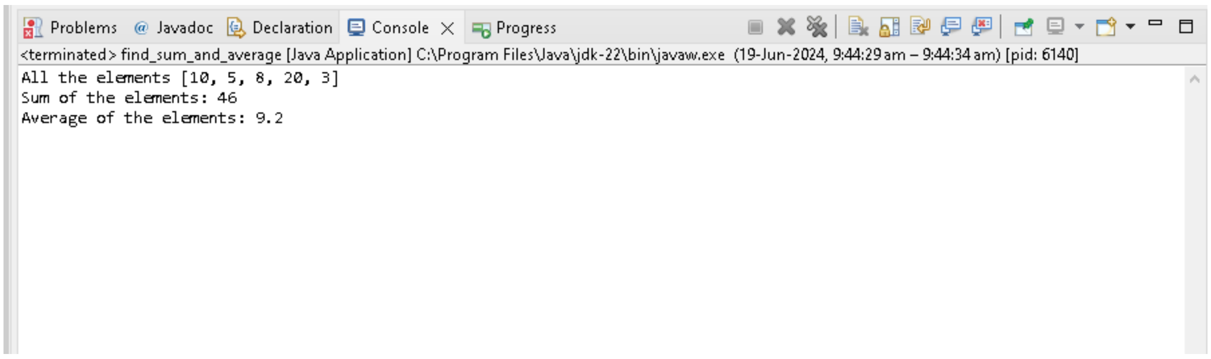
```
<terminated> find_sum_and_average [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (19-Jun-2024, 9:44:29 am – 9:44:34 am) [pid: 6140]
All the elements [10, 5, 8, 20, 3]
Sum of the elements: 46
Average of the elements: 9.2
```

30 : Reverse each word of a string using Java 8 streams

```
package java8;
import java.util.Arrays;
import java.util.Scanner;
import java.util.stream.Collectors;

public class Reverse_string {

        public static void main(String[] args)
        {
            System.out.println("Enter the name");
            Scanner obj=new Scanner(System.in);
            String input=obj.nextLine();


           String reversed = Arrays.stream(input.split("\\s+"))
```

```
                        .map(word -> new
StringBuilder(word).reverse().toString())
                        .collect(Collectors.joining(" "));

          System.out.println("Original string: " + input);
          System.out.println("String with reversed words: " +
reversed);
      }
  }
```

OUTPUT



```
<terminated> find_sum_and_average [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe  (19-Jun-2024, 9:44:29 am – 9:44:34 am) [pid: 6140]
All the elements [10, 5, 8, 20, 3]
Sum of the elements: 46
Average of the elements: 9.2
```