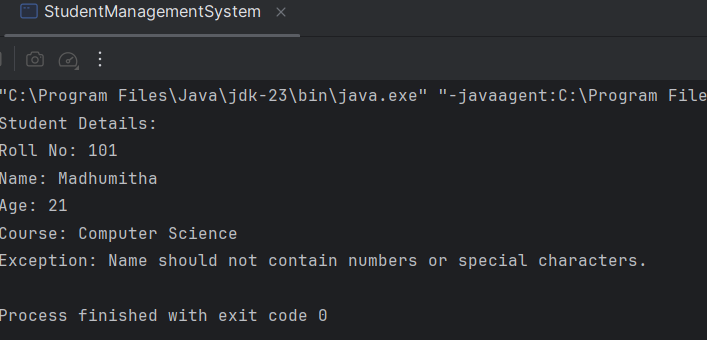
TASK-4

1. Ramesh is developing a student management system for a university. In this system, you have a Student class to represent student information. You are asked to help Ramesh to handle exception which can be occurred into program according to following Scenarios:

* class Student with attributes roll no, name, age and course. Initialize values through parameterized constructors.
* If the age of the student is not between 15 and 21 then generate a user-defined exception "AgeNotWithinRangeException".
* If a name contains numbers or special symbols, raise exception "NameNot ValidException". Define the two exception classes.

package advanced\_Java;  
//Custom Exception for Age Validation  
class AgeNotWithinRangeException extends Exception {  
 public AgeNotWithinRangeException(String message) {  
 super(message);  
 }  
}  
//Custom Exception for Name Validation  
class NameNotValidException extends Exception {  
 public NameNotValidException(String message) {  
 super(message);  
 }  
}  
//Student Class  
class Student {  
 private int rollNo;  
 private String name;  
 private int age;  
 private String course;  
 // Constructor with exception handling  
 public Student(int rollNo, String name, int age, String course)  
 throws AgeNotWithinRangeException, NameNotValidException {  
 this.rollNo = rollNo;  
 this.course = course;  
 // Validate age  
 if (age < 15 || age > 21) {  
 throw new AgeNotWithinRangeException("Age must be between 15 and 21.");  
 }  
 this.age = age;  
 // Validate name  
 if (!name.matches("[a-zA-Z ]+")) { // Name should only contain letters and spaces  
 throw new NameNotValidException("Name should not contain numbers or special characters.");  
 }  
 this.name = name;  
 }  
 // Method to display student details  
 public void displayStudentInfo() {  
 System.*out*.println("Student Details:");  
 System.*out*.println("Roll No: " + rollNo);  
 System.*out*.println("Name: " + name);  
 System.*out*.println("Age: " + age);  
 System.*out*.println("Course: " + course);  
 }  
}  
public class StudentManagementSystem {  
 public static void main(String[] args) {  
 // *TODO Auto-generated method stub* try {  
 // Creating a valid student  
 Student student1 = new Student(101, "Madhumitha", 21, "Computer Science");  
 student1.displayStudentInfo();  
 // Creating a student with an invalid age  
 Student student2 = new Student(102, "Shivani Gupta", 15, "Mathematics");  
 // Creating a student with an invalid name  
 Student student3 = new Student(103, "Ajay@123", 19, "Physics");  
 } catch (AgeNotWithinRangeException | NameNotValidException e) {  
 System.*out*.println("Exception: " + e.getMessage());  
 }  
 }  
}

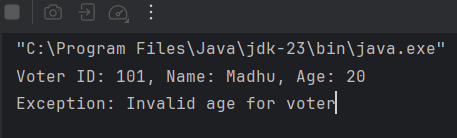
OUTPUT:



1. Create a class Voter(voterld, name, age) with parameterized constructor. The parameterized constructor should throw a checked/Unchecked exception if age is less than 18. The message of exception is "invalid age for voter"

package advanced\_Java;  
// Custom Exception Class  
public class Task\_4 extends Exception {  
 public Task\_4(String message) {  
 super(message);  
 }  
}  
// Voter Class  
class Voter {  
 private int voterId;  
 private String name;  
 private int age;  
 // Parameterized constructor  
 public Voter(int voterId, String name, int age) throws Task\_4 {  
 if (age < 18) {  
 throw new Task\_4("Invalid age for voter"); // Throwing custom checked exception  
 }  
 this.voterId = voterId;  
 this.name = name;  
 this.age = age;  
 }  
 @Override  
 public String toString() {  
 return "Voter ID: " + voterId + ", Name: " + name + ", Age: " + age;  
 }  
 // Main method to test the Voter class  
 public static void main(String[] args) {  
 try {  
 Voter v1 = new Voter(101, "Madhu", 20);  
 System.*out*.println(v1);  
 Voter v2 = new Voter(102, "Nirmal", 16); // This will throw an exception  
 System.*out*.println(v2);  
 } catch (Task\_4 e) { // Catching the correct custom exception  
 System.*out*.println("Exception: " + e.getMessage());  
 }  
 }  
}

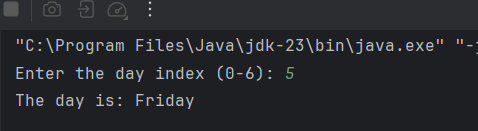
OUTPUT:



1. Store name of weekdays in an array (starting from "Sunday" at 0 index). Ask day position from user and print day name. Handle array index out of bound exception and give proper message if user enters day index outside range (0-6).

package advanced\_Java;  
import java.util.Scanner;  
public class WeekdayFinder {  
 public static void main(String[] args) {  
 // Array to store weekdays (index starts from 0 for Sunday)  
 String[] weekdays = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"};  
 // Scanner object to take user input  
 Scanner scanner = new Scanner(System.*in*);  
 try {  
 // Asking user for input  
 System.*out*.print("Enter the day index (0-6): ");  
 int dayIndex = scanner.nextInt();  
 // Printing the corresponding weekday  
 System.*out*.println("The day is: " + weekdays[dayIndex]);  
 } catch (ArrayIndexOutOfBoundsException e) {  
 // Handling index out of range  
 System.*out*.println("Error: Invalid index! Please enter a number between 0 and 6.");  
 } catch (Exception e) {  
 // Handling other unexpected exceptions (e.g., input mismatch)  
 System.*out*.println("Error: Invalid input! Please enter a valid number.");  
 } finally {  
 scanner.close(); // Closing the scanner  
 }  
 }  
}

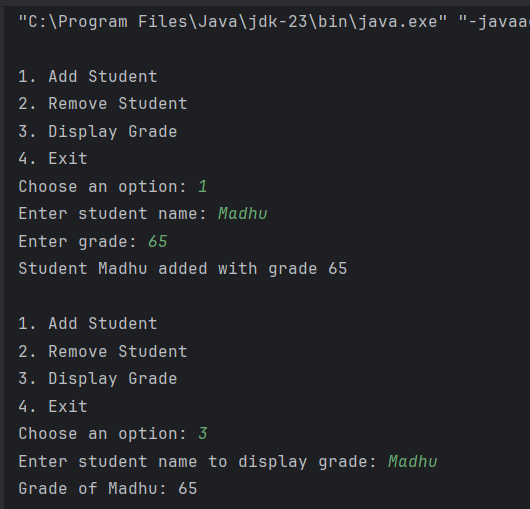
OUTPUT:

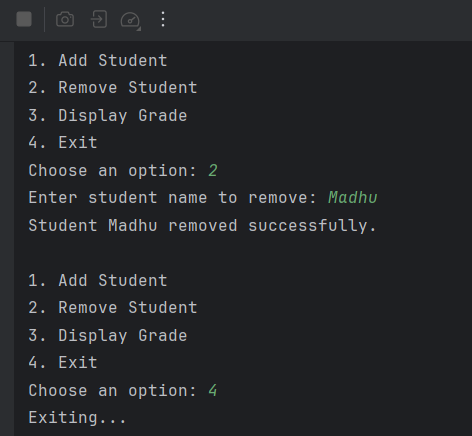


1. Create a HashMap where keys are student names (strings) and values are their corresponding grades (integers). Create methods to add a new student, remove a student, and Display up a student's grade by name.

package advanced\_Java;  
import java.util.HashMap;  
import java.util.Scanner;  
public class StudentGrades {  
 private HashMap<String, Integer> studentMap;  
 // Constructor to initialize HashMap  
 public StudentGrades() {  
 studentMap = new HashMap<>();  
 }  
 // Method to add a student  
 public void addStudent(String name, int grade) {  
 studentMap.put(name, grade);  
 System.*out*.println("Student " + name + " added with grade " + grade);  
 }  
 // Method to remove a student  
 public void removeStudent(String name) {  
 if (studentMap.containsKey(name)) {  
 studentMap.remove(name);  
 System.*out*.println("Student " + name + " removed successfully.");  
 } else {  
 System.*out*.println("Student " + name + " not found.");  
 }  
 }  
 // Method to display a student's grade  
 public void displayGrade(String name) {  
 if (studentMap.containsKey(name)) {  
 System.*out*.println("Grade of " + name + ": " + studentMap.get(name));  
 } else {  
 System.*out*.println("Student " + name + " not found.");  
 }  
 }  
 // Main method to test the functionality  
 public static void main(String[] args) {  
 StudentGrades sg = new StudentGrades();  
 Scanner scanner = new Scanner(System.*in*);  
 while (true) {  
 System.*out*.println("\n1. Add Student\n2. Remove Student\n3. Display Grade\n4. Exit");  
 System.*out*.print("Choose an option: ");  
 int choice = scanner.nextInt();  
 scanner.nextLine(); // Consume newline  
 switch (choice) {  
 case 1:  
 System.*out*.print("Enter student name: ");  
 String name = scanner.nextLine();  
 System.*out*.print("Enter grade: ");  
 int grade = scanner.nextInt();  
 sg.addStudent(name, grade);  
 break;  
 case 2:  
 System.*out*.print("Enter student name to remove: ");  
 name = scanner.nextLine();  
 sg.removeStudent(name);  
 break;  
 case 3:  
 System.*out*.print("Enter student name to display grade: ");  
 name = scanner.nextLine();  
 sg.displayGrade(name);  
 break;  
 case 4:  
 System.*out*.println("Exiting...");  
 scanner.close();  
 return;  
 default:  
 System.*out*.println("Invalid choice! Please try again.");  
 }  
 }  
 }  
}

OUTPUT:





1. Use Collection Classes to store Integers Create some methods for following functionalities. a. Include functions for pushing elements onto the stack. b. popping elements from the stack.

package advanced\_Java;  
import java.util.Stack;  
import java.util.Scanner;  
public class IntegerStack {  
 private Stack<Integer> stack;  
 // Constructor to initialize the stack  
 public IntegerStack() {  
 stack = new Stack<>();  
 }  
 // Method to push an element onto the stack  
 public void pushElement(int element) {  
 stack.push(element);  
 System.*out*.println(element + " pushed onto the stack.");  
 }  
 // Method to pop an element from the stack  
 public void popElement() {  
 if (stack.isEmpty()) {  
 System.*out*.println("Stack is empty! Cannot pop elements.");  
 } else {  
 int removedElement = stack.pop();  
 System.*out*.println("Popped element: " + removedElement);  
 }  
 }  
 // Method to display the current stack  
 public void displayStack() {  
 if (stack.isEmpty()) {  
 System.*out*.println("Stack is empty.");  
 } else {  
 System.*out*.println("Current Stack: " + stack);  
 }  
 }  
 // Main method to test the stack operations  
 public static void main(String[] args) {  
 IntegerStack intStack = new IntegerStack();  
 Scanner scanner = new Scanner(System.*in*);  
 while (true) {  
 System.*out*.println("\n1. Push Element");  
 System.*out*.println("2. Pop Element");  
 System.*out*.println("3. Display Stack");  
 System.*out*.println("4. Exit");  
 System.*out*.print("Choose an option: ");  
 int choice = scanner.nextInt();  
 switch (choice) {  
 case 1:  
 System.*out*.print("Enter an integer to push: ");  
 int element = scanner.nextInt();  
 intStack.pushElement(element);  
 break;  
 case 2:  
 intStack.popElement();  
 break;  
 case 3:  
 intStack.displayStack();  
 break;  
 case 4:  
 System.*out*.println("Exiting...");  
 scanner.close();  
 return;  
 default:  
 System.*out*.println("Invalid choice! Please try again.");  
 }  
 }  
 }  
}

OUTPUT:

