# AIM:

Write a program using Dart to collects user input (name, age, and favourite colour), Stores this input in variables, it uses conditional statements to print personalised messages based on age, Organises the collected data into collections (List, Set, and Map), Uses Dart operators and constants.

# THEORY:

# The provided code demonstrates the use of Object-Oriented Programming (OOP) and functional programming concepts in Dart. It highlights several features of the Dart language, such as mixins, inheritance, encapsulation, collections, and user interaction.

# Key Concepts in Dart Highlighted in the Code:

# Classes and Objects:

# Student is a base class.

# Alumi inherits from Student and mixes in Plasmentstudent and Internshipstudent.

# Mixins:

# Plasmentstudent and Internshipstudent are declared as mixin to allow code reuse across multiple classes.

# Alumi uses these mixins to inherit properties and methods.

# Encapsulation:

# The Student class uses private properties (Rollno, name, marks) with get and set methods to encapsulate data.

# Inheritance:

# The Alumi class extends the Student class and combines additional functionality using mixins.

# Input and Output:

# The stdin and stdout modules are used to take user input and display outputs.

# Control Structures:

# if-else is used for eligibility checks.

# A while loop manages the menu-driven system.

# switch-case handles various menu options.

# List Operations:

# A List<Student> is used to store multiple Student objects.

# Iteration and searching are implemented using loops.

# Functionality:

# A standalone function s(int i) demonstrates functional programming by squaring a number.

**CODE:**

import 'dart:io';

import 'package:myapp/myapp.dart' as myapp;

class Student{

    int Rollno=0;

    String name ="";

    double marks=0.0;

    Student(){}

     void setRollno(int Rollno){

       this.Rollno = Rollno;

     }

     void setName(String name){

       this.name = name;

     }

     void setMarks(double marks){

       this.marks = marks;

     }

     String getName(){

       return name;

     }

      int getRollno(){

        return Rollno;

      }

      double getMarks(){

        return marks;

      }

}

mixin Plasmentstudent {

  String company ="";

  double salary = 0.0;

  void setCompany(String company){

    this.company = company;

  }

  void setSalary(double salary){

    this.salary = salary;

  }

  String getCompany(){

    return company;

  }

  double getSalary(){

    return salary;

  }

}

mixin Internshipstudent{

  String company ="";

  double stipend = 0.0;

  void setCompany(String company){

    this.company = company;

  }

  void setStipend(double stipend){

    this.stipend = stipend;

  }

  String getCompany(){

    return company;

  }

  double getStipend(){

    return stipend;

  }

}

class Alumi extends Student with Plasmentstudent,Internshipstudent{

  Alumi(){}

}

int s(int i){

  return i\*i;

}

void main(List<String> arguments) {

  print('Hello world: ${myapp.calculate()}!');

  print("Enter your name : ");

  String ? name = stdin.readLineSync();

  print("Hello $name");

  print("Enter your age : ");

  int  ? age = int.parse(stdin.readLineSync()!);

  if(age < 18){

    print("You are not eligible to vote");

  }else{

    print("You are eligible to vote");

  }

   List<Student> a = <Student>[];

 int  c =0;

 while(c!=6){

  print("1. Add Student");

  print("2. Display Student");

  print("3. Search Student");

  print("4. Delete Student");

  print("5. Update Student");

  print("6. Exit");

   c = int.parse(stdin.readLineSync()!);

  switch(c)

  {

    case 1:

  Student s = new Student();

  s.setName( stdin.readLineSync()!);

  s.setRollno( int.parse(stdin.readLineSync()!));

  s.setMarks(double.parse(stdin.readLineSync()!));

  print("Name : ${s.getName()}");

  print("Rollno : ${s.getRollno()}");

  print("Marks : ${s.getMarks()}");

  a.add(s);

     break;

    case 2:

    for(int i=0;i<a.length;i++){

      print("Name : ${a[i].getName()}");

      print("Rollno : ${a[i].getRollno()}");

      print("Marks : ${a[i].getMarks()}");

    }

    break;

    case 3:

    print("Enter the rollno to search : ");

    int ? rollno = int.parse(stdin.readLineSync()!);

    for(int i=0;i<a.length;i++){

      if(a[i].getRollno() == rollno){

        print("Name : ${a[i].getName()}");

        print("Rollno : ${a[i].getRollno()}");

        print("Marks : ${a[i].getMarks()}");

      }

    }

    break;

    case 4:

    print("Enter the rollno to delete : ");

    int ? rollno = int.parse(stdin.readLineSync()!);

    for(int i=0;i<a.length;i++){

      if(a[i].getRollno() == rollno){

        a.removeAt(i);

      }

    }

    break;

    case 5:

    print("Enter the rollno to update : ");

    int ? rollno = int.parse(stdin.readLineSync()!);

    for(int i=0;i<a.length;i++){

      if(a[i].getRollno() == rollno){

        print("Enter the name : ");

        a[i].setName(stdin.readLineSync()!);

        print("Enter the rollno : ");

        a[i].setRollno(int.parse(stdin.readLineSync()!));

        print("Enter the marks : ");

        a[i].setMarks(double.parse(stdin.readLineSync()!));

      }

    }

    break;

    default : print("Invalid choice");

  }

  }

  int i = stdin.readLineSync() as int;

  print(s(i));

}

**OUTPUT:**

Here, we enter name, age then we get our output like this.

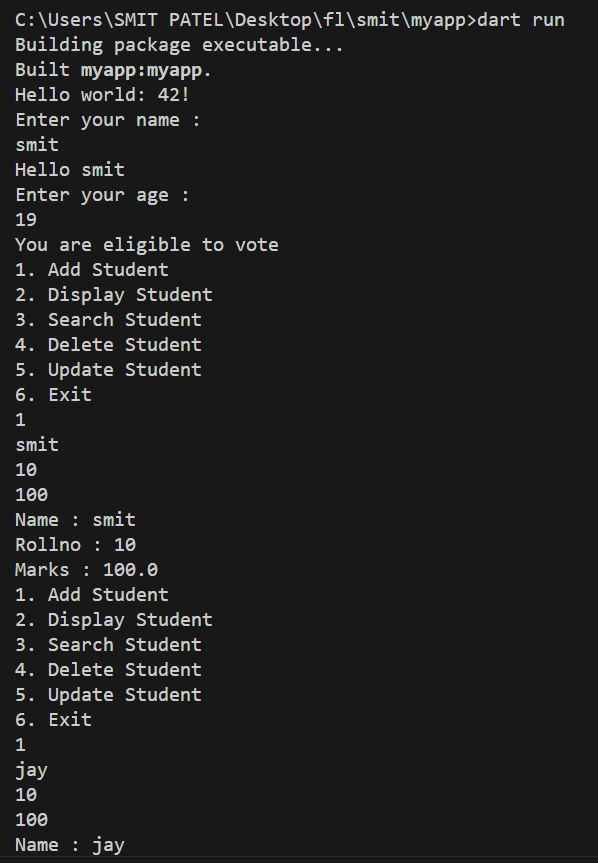


Figure 1: Entering the appropriate values

Now, we enter the values which are inappropriate and the program shows the error.

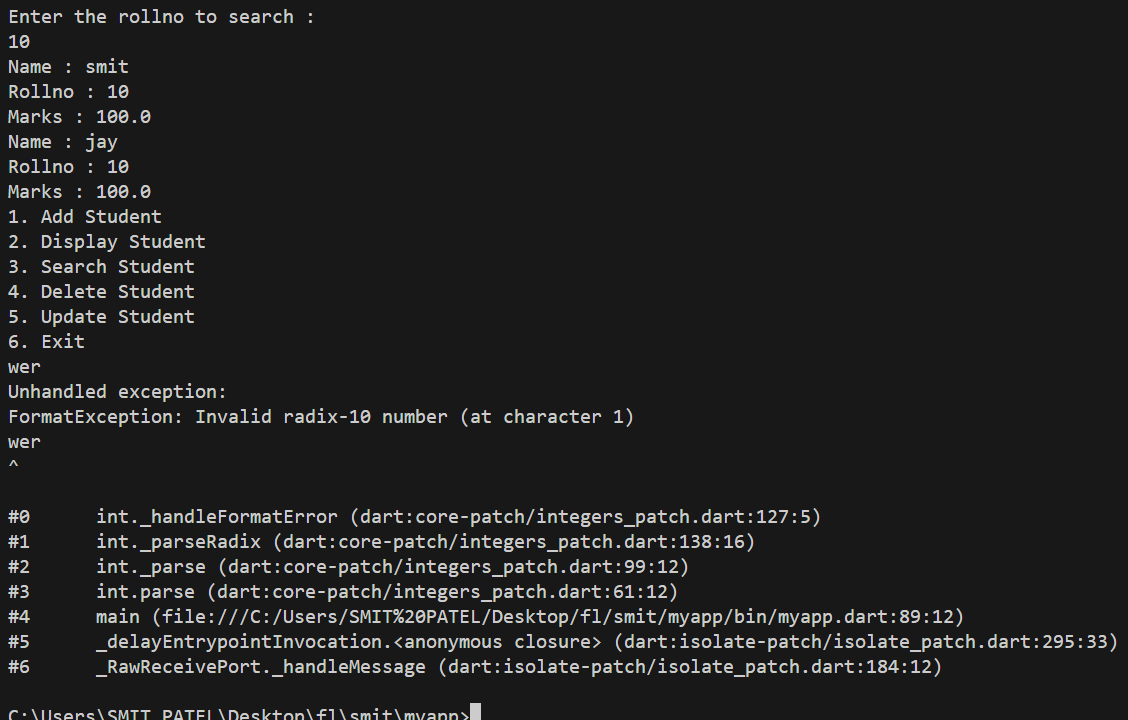


Figure 2:Entering the inappropriate values

# Latest Applications:

1. **Cross-Platform Development:**
   * Dart is the primary language for Flutter, which is used for building cross-platform mobile, web, and desktop applications.
   * Example: Building dynamic and responsive UIs.
2. **Backend Development:**
   * Dart can be used for server-side development using frameworks like **Aqueduct** or **Dart Frog**.
3. **Web Development:**
   * Dart powers web applications using **Dart2js**, which compiles Dart to JavaScript.
4. **Machine Learning:**
   * Libraries like **TensorFlow Dart** are emerging, enabling Dart to be used in ML applications.
5. **IoT Applications:**
   * Dart's efficiency and asynchronous nature make it suitable for lightweight IoT applications

# Learning Outcome:

* **Understanding Dart Fundamentals:**
  + Mastering Dart syntax, data types, control structures, and OOP concepts.
* **Proficiency in Flutter:**
  + Learning to create cross-platform UIs and interact with various APIs.
* **Mixins and Reusability:**
  + Understanding how to use mixins for clean and reusable code.
* **Efficient Memory and State Management:**
  + Learning how Dart handles asynchronous programming using Future, Stream, and await.
* **Versatility Across Platforms:**
  + Gaining experience in using Dart for mobile, web, and server-side applications.
* **Scalability and Performance:**
  + Building applications that scale efficiently while maintaining high performance.
* Mathematical calculations: Performing basic calculations (e.g., calculating years until adulthood) and using these results to generate dynamic output.
* Extensibility: Designing the code to be easily extendable in the future with new features or functionality.