VD1

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

public class GreetingServer {

    public static void main (String [] args){

        try {

            GreetingService greetingService = new GreetingServiceImpl();

            Registry registry = LocateRegistry.createRegistry(1099);

            registry.rebind("GreetingService", greetingService);

            System.out.println("Server is running....");

        } catch (Exception e)

        {

            e.printStackTrace();

        }

    }

}

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

public class GreetingClient {

    public static void main(String [] args){

        try{

            Registry registry = LocateRegistry.getRegistry("localhost",1099);

            GreetingService greetingService = (GreetingService) registry.lookup("GreetingService");

            String response = greetingService.sayHello("John");

            System.out.println(response);

        }catch(Exception e){

            e.printStackTrace();

        }

    }

}

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface GreetingService extends Remote {

    String sayHello(String name) throws RemoteException;

}

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

public class GreetingServiceImpl extends UnicastRemoteObject implements GreetingService {

  public GreetingServiceImpl() throws RemoteException{

    super();

  }

  @Override

  public String sayHello(String name) throws RemoteException{

    return "Hello, " + name + "!";

  }

}

A screenshot of a computer

Description automatically generated

VD2

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

public class Server {

    public static void main (String [] args ){

        try{

            Calculator calculator=new CalculatorImpl();

       //     Calculator stub = (Calculator) UnicastRemoteObject.exportObject(calculator, 0);

       Registry registry;

        try{

              registry = LocateRegistry.getRegistry(1099);

              registry.list();

        }catch (Exception e){

            registry=LocateRegistry.createRegistry(1099);

        }

        registry.rebind("CalculatorService", calculator);

        System.out.println("Server is ready");

        }catch(Exception e)

        {

            e.printStackTrace();

        }

    }

}

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

public class Client {

    public static void main (String [] args ){

        try{

            Registry registry = LocateRegistry.getRegistry(1099);

            Calculator calculator = (Calculator) registry.lookup("CalculatorService");

            int sum= calculator.add(5,3);

            int difference= calculator.subtract(7, 2);

            int product = calculator.multiply(4,6);

            int quotient = calculator.divide(10, 2);

            System.out.println("Sum:" + sum);

            System.out.println("Difference:" + difference);

            System.out.println("Product:" + product);

            System.out.println("Quotient:" + quotient);

        }catch (Exception e)

        {

            e.printStackTrace();

        }

    }

}

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface Calculator extends Remote {

    int add (int a, int b) throws RemoteException;

    int subtract (int a, int b) throws RemoteException;

    int multiply (int a, int b) throws RemoteException;

    int divide (int a, int b) throws RemoteException;

}

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

public class CalculatorImpl extends UnicastRemoteObject implements Calculator {

    protected CalculatorImpl() throws RemoteException{

        super();

    }

    @Override

    public int add(int a,int b) throws RemoteException{

        return a+b;

    }

    @Override

    public int subtract(int a,int b)throws RemoteException{

        return a - b;

    }

    public int multiply (int a, int b) throws RemoteException{

        return a \* b;

    }

    public int divide (int a, int b) throws RemoteException{

        return a / b;

    }

}

A screenshot of a computer

Description automatically generated

Bài 1

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

public class StudentServer {

    public static void main(String[] args) {

        try {

            StudentManagerImpl studentManager = new StudentManagerImpl();

            Registry registry = LocateRegistry.createRegistry(1099);

            registry.rebind("StudentManager", studentManager);

            System.out.println("Student Manager is ready.");

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

}

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import java.util.List;

import java.util.Scanner;

public class StudentClient {

    public static void main(String[] args) {

        try {

            Registry registry = LocateRegistry.getRegistry("localhost", 1099);

            StudentManager studentManager = (StudentManager) registry.lookup("StudentManager");

            Scanner scanner = new Scanner(System.in);

            while (true) {

                System.out.println("1. Add Student");

                System.out.println("2. List Students");

                System.out.println("3. Exit");

                System.out.print("Choose an option: ");

                int choice = scanner.nextInt();

                scanner.nextLine(); // Consume newline

                switch (choice) {

                    case 1:

                        System.out.print("Enter ID: ");

                        String id = scanner.nextLine();

                        System.out.print("Enter Name: ");

                        String name = scanner.nextLine();

                        System.out.print("Enter Age: ");

                        int age = scanner.nextInt();

                        studentManager.addStudent(id, name, age);

                        break;

                    case 2:

                        List<Student> students = studentManager.getStudentList();

                        for (Student student : students) {

                            System.out.println(student);

                        }

                        break;

                    case 3:

                        System.exit(0);

                        break;

                    default:

                        System.out.println("Invalid option. Try again.");

                        break;

                }

            }

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

}

import java.io.Serializable;

public class Student implements Serializable {

    private String id;

    private String name;

    private int age;

    public Student(String id, String name, int age) {

        this.id = id;

        this.name = name;

        this.age = age;

    }

    // Getters and toString() method

    public String getId() {

        return id;

    }

    public String getName() {

        return name;

    }

    public int getAge() {

        return age;

    }

    @Override

    public String toString() {

        return "Student{id='" + id + "', name='" + name + "', age=" + age + "}";

    }

}

import java.rmi.Remote;

import java.rmi.RemoteException;

import java.util.List;

public interface StudentManager extends Remote {

    void addStudent(String id, String name, int age) throws RemoteException;

    List<Student> getStudentList() throws RemoteException;

}

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

public class StudentManagerImpl extends UnicastRemoteObject implements StudentManager {

    private Map<String, Student> students;

    protected StudentManagerImpl() throws RemoteException {

        students = new HashMap<>();

    }

    @Override

    public void addStudent(String id, String name, int age) throws RemoteException {

        students.put(id, new Student(id, name, age));

    }

    @Override

    public List<Student> getStudentList() throws RemoteException {

        return new ArrayList<>(students.values());

    }

}

A screenshot of a computer

Description automatically generated

Bài 2

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import java.util.List;

import java.util.Scanner;

public class Bai2Client {

    public static void main(String[] args) {

        try {

            Registry registry = LocateRegistry.getRegistry("localhost", 1099);

            BookManager bookManager = (BookManager) registry.lookup("BookManager");

            Scanner scanner = new Scanner(System.in);

            while (true) {

                System.out.println("Menu:");

                System.out.println("1. Xem danh sách tất cả sách");

                System.out.println("2. Tìm kiếm sách theo tiêu đề");

                System.out.println("3. Tìm kiếm sách theo tác giả");

                System.out.println("4. Thêm sách mới");

                System.out.println("5. Cập nhật thông tin sách");

                System.out.println("6. Xóa sách theo ID");

                System.out.println("7. Thoát");

                int choice = scanner.nextInt();

                scanner.nextLine(); // Consume newline

                switch (choice) {

                    case 1:

                        List<Book> allBooks = bookManager.getAllBooks();

                        allBooks.forEach(System.out::println);

                        break;

                    case 2:

                        System.out.print("Nhập từ khóa tiêu đề: ");

                        String titleKeyword = scanner.nextLine();

                        List<Book> booksByTitle = bookManager.searchBooksByTitle(titleKeyword);

                        booksByTitle.forEach(System.out::println);

                        break;

                    case 3:

                        System.out.print("Nhập từ khóa tác giả: ");

                        String authorKeyword = scanner.nextLine();

                        List<Book> booksByAuthor = bookManager.searchBooksByAuthor(authorKeyword);

                        booksByAuthor.forEach(System.out::println);

                        break;

                    case 4:

                        System.out.print("Nhập ID sách: ");

                        int id = scanner.nextInt();

                        scanner.nextLine(); // Consume newline

                        System.out.print("Nhập tiêu đề sách: ");

                        String title = scanner.nextLine();

                        System.out.print("Nhập tác giả sách: ");

                        String author = scanner.nextLine();

                        bookManager.addBook(new Book(id, title, author));

                        break;

                    case 5:

                        System.out.print("Nhập ID sách cần cập nhật: ");

                        int updateId = scanner.nextInt();

                        scanner.nextLine(); // Consume newline

                        System.out.print("Nhập tiêu đề mới: ");

                        String newTitle = scanner.nextLine();

                        System.out.print("Nhập tác giả mới: ");

                        String newAuthor = scanner.nextLine();

                        bookManager.updateBook(new Book(updateId, newTitle, newAuthor));

                        break;

                    case 6:

                        System.out.print("Nhập ID sách cần xóa: ");

                        int deleteId = scanner.nextInt();

                        bookManager.deleteBook(deleteId);

                        break;

                    case 7:

                        System.out.println("Thoát chương trình");

                        scanner.close();

                        return;

                }

            }

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

}

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

public class Bai2Server {

    public static void main(String[] args) {

        try {

            BookManagerImpl bookManager = new BookManagerImpl();

            Registry registry = LocateRegistry.createRegistry(1099);

            registry.rebind("BookManager", bookManager);

            System.out.println("Server is running...");

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

}

import java.io.Serializable;

public class Book implements Serializable {

    private int id;

    private String title;

    private String author;

    public Book(int id, String title, String author) {

        this.id = id;

        this.title = title;

        this.author = author;

    }

    public int getId() {

        return id;

    }

    public String getTitle() {

        return title;

    }

    public String getAuthor() {

        return author;

    }

    public void setTitle(String title) {

        this.title = title;

    }

    public void setAuthor(String author) {

        this.author = author;

    }

    @Override

    public String toString() {

        return "Book [id=" + id + ", title=" + title + ", author=" + author + "]";

    }

}

import java.rmi.Remote;

import java.rmi.RemoteException;

import java.util.List;

public interface BookManager extends Remote {

    List<Book> getAllBooks() throws RemoteException;

    Book getBookById(int id) throws RemoteException;

    List<Book> searchBooksByTitle(String keyword) throws RemoteException;

    List<Book> searchBooksByAuthor(String keyword) throws RemoteException;

    void addBook(Book book) throws RemoteException;

    void updateBook(Book book) throws RemoteException;

    void deleteBook(int id) throws RemoteException;

}

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

import java.util.ArrayList;

import java.util.List;

import java.util.stream.Collectors;

public class BookManagerImpl extends UnicastRemoteObject implements BookManager {

    private List<Book> books;

    protected BookManagerImpl() throws RemoteException {

        books = new ArrayList<>();

    }

    @Override

    public List<Book> getAllBooks() throws RemoteException {

        return books;

    }

    @Override

    public Book getBookById(int id) throws RemoteException {

        return books.stream().filter(book -> book.getId() == id).findFirst().orElse(null);

    }

    @Override

    public List<Book> searchBooksByTitle(String keyword) throws RemoteException {

        return books.stream().filter(book -> book.getTitle().contains(keyword)).collect(Collectors.toList());

    }

    @Override

    public List<Book> searchBooksByAuthor(String keyword) throws RemoteException {

        return books.stream().filter(book -> book.getAuthor().contains(keyword)).collect(Collectors.toList());

    }

    @Override

    public void addBook(Book book) throws RemoteException {

        books.add(book);

    }

    @Override

    public void updateBook(Book book) throws RemoteException {

        Book existingBook = getBookById(book.getId());

        if (existingBook != null) {

            existingBook.setTitle(book.getTitle());

            existingBook.setAuthor(book.getAuthor());

        }

    }

    @Override

    public void deleteBook(int id) throws RemoteException {

        books.removeIf(book -> book.getId() == id);

    }

}

A screenshot of a computer

Description automatically generated