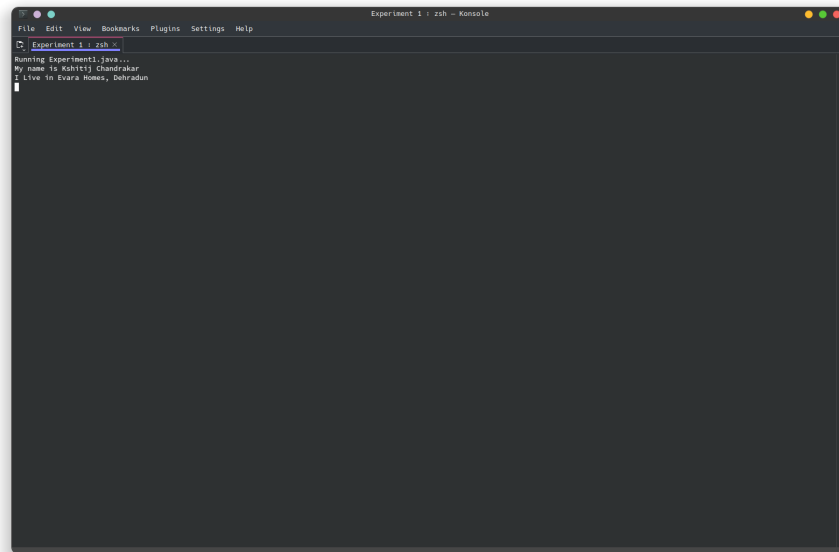


./Experiment 1

./Experiment 1/Experiment1.java

Code

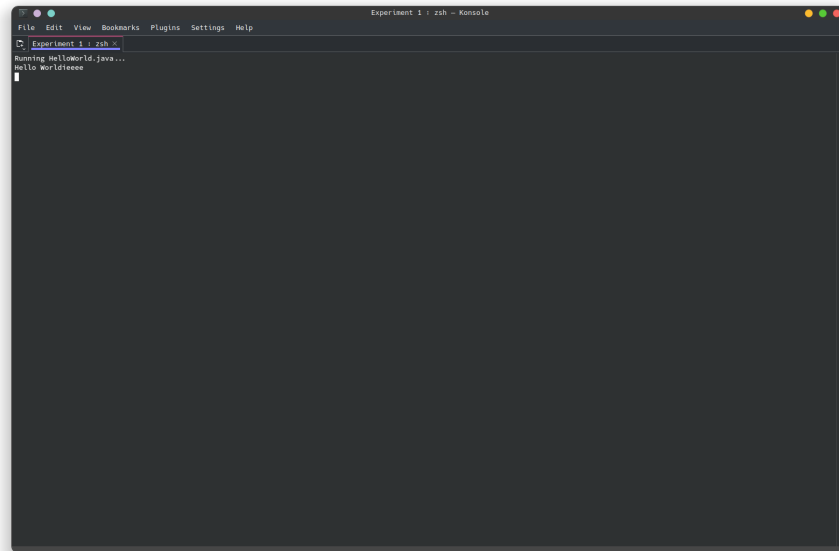
```
class Experiment1{  
    public static void main (String[] args){  
        System.out.println("My name is Kshitij Chandrakar");  
        System.out.println("I Live in Evara Homes, Dehradun");  
    }  
}
```



Output

./Experiment 1/HelloWorld.java #### Code

```
class test{  
    public static void main (String[] args){  
        System.out.println("Hello Worldieeee");  
    }  
}
```



Output

./Experiment 10 ### ./Experiment 10/Employee.java ##### Code

```
import java.util.ArrayList;
import java.util.Iterator;

class Employee {
    private String name;
    private int id;
    private double salary;

    // Constructor
    public Employee(String name, int id, double salary) {
        this.name = name;
        this.id = id;
        this.salary = salary;
    }

    // Getters and Setters
    public String getName() {
        return name;
    }

    public int getId() {
        return id;
    }

    public double getSalary() {
```

```

        return salary;
    }

    public void setSalary(double salary) {
        this.salary = salary;
    }

    @Override
    public String toString() {
        return "Employee [Name: " + name + ", ID: " + id + ", Salary: $" + salary + "]\n";
    }
}

public class EmployeeManagement {
    public static void main(String[] args) {
        // Create ArrayList to store employees
        ArrayList<Employee> employees = new ArrayList<>();

        // Add three employees
        employees.add(new Employee("John Doe", 101, 50000.0));
        employees.add(new Employee("Jane Smith", 102, 60000.0));
        employees.add(new Employee("Bob Johnson", 103, 55000.0));

        System.out.println("Initial Employee List:");
        printEmployees(employees);

        // Update Jane's salary (ID 102)
        for (Employee emp : employees) {
            if (emp.getId() == 102) {
                emp.setSalary(65000.0);
                System.out.println("\nUpdated salary for Jane Smith (ID 102)");
                break;
            }
        }

        // Remove employee with ID 101 (John Doe)
        Iterator<Employee> iterator = employees.iterator();
        while (iterator.hasNext()) {
            Employee emp = iterator.next();
            if (emp.getId() == 101) {
                iterator.remove();
                System.out.println("Removed employee with ID 101");
                break;
            }
        }
    }
}

```

```

        System.out.println("\nFinal Employee List:");
        printEmployees(employees);
    }

    // Helper method to print all employees
    private static void printEmployees(ArrayList<Employee> employees) {
        if (employees.isEmpty()) {
            System.out.println("No employees in the list");
            return;
        }

        for (Employee emp : employees) {
            System.out.println(emp);
        }
    }
}

```

Output ./Experiment 10/Employee.java ### ./Experiment 10/PrimeNumberChecker.java ##### Code

```

import java.util.ArrayList;

public class PrimeNumberChecker {
    public static void main(String[] args) {
        ArrayList<Integer> numbers = new ArrayList<>();
        numbers.add(2);
        numbers.add(3);
        numbers.add(4);
        numbers.add(5);
        numbers.add(6);
        numbers.add(7);
        numbers.add(8);
        numbers.add(9);
        numbers.add(10);
        numbers.add(11);
        numbers.add(12);
        numbers.add(13);

        for (Integer num : numbers) {
            int n = num; // Unboxing
            boolean isPrime = isPrimeNumber(n);
            System.out.println(n + " is " + (isPrime ? "prime" : "not prime"));
        }
    }

    private static boolean isPrimeNumber(int number) {

```

```

    if (number <= 1) {
        return false;
    }
    if (number == 2) {
        return true;
    }
    if (number % 2 == 0) {
        return false;
    }
    for (int i = 3; i * i <= number; i += 2) {
        if (number % i == 0) {
            return false;
        }
    }
    return true;
}
}

```

Output ./Experiment 10/PrimeNumberChecker.java ## ./Experiment 2
 ### ./Experiment 2/Calculator.java ##### Code

```

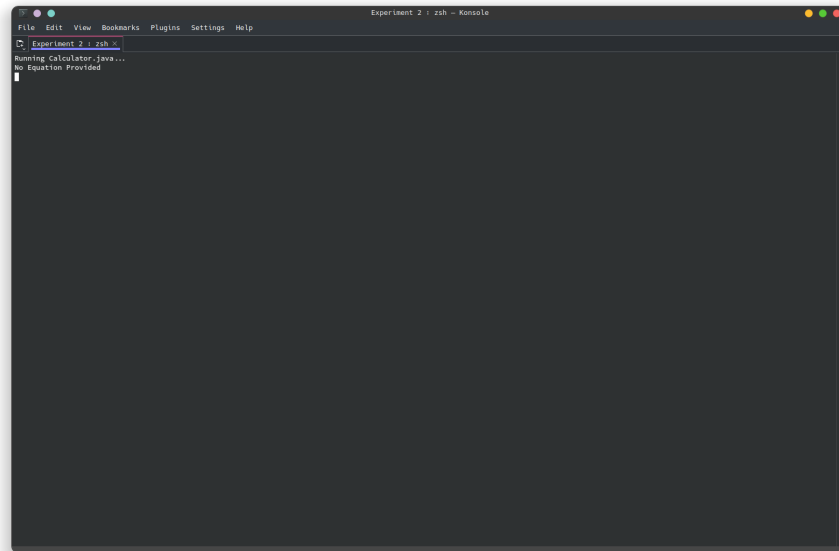
class Calculator {
    public static void main(String[] args) {
        // Check if length of args array is greater than 0
        if (args.length <= 0) {
            System.out.println("No Equation Provided");
        } else {
            String equation = String.join("", args);
            float result = 0;
            String operand = "";
            char lastOperator = '+';
            for (int i = 0; i < equation.length(); i++) {
                char ch = equation.charAt(i);
                if (Character.isDigit(ch) || ch == '.') {
                    operand = operand + ch;
                } else {
                    if (!operand.isEmpty()) {
                        float num = Float.parseFloat(operand);
                        switch (lastOperator) {
                            case '+':
                                result += num;
                                break;
                            case '-':
                                result -= num;
                                break;
                            case '*':

```

```

        result *= num;
        break;
    case '/':
        result /= num;
        break;
    }
}
operand = "";
lastOperator = ch;
}
}
if (!operand.isEmpty()) {
    float num = Float.parseFloat(operand);
    switch (lastOperator) {
        case '+':
            result += num;
            break;
        case '-':
            result -= num;
            break;
        case '*':
            result *= num;
            break;
        case '/':
            result /= num;
            break;
    }
}
System.out.println(result);
}
}
}
}

```

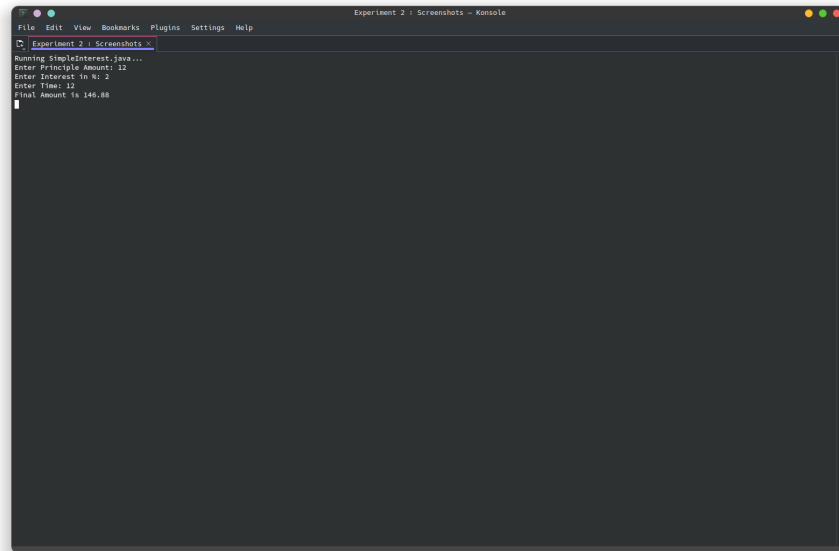


Output

./Experiment 2/SimpleInterest.java #### Code

```
import java.util.Scanner; // Import the Scanner class

class SimpleInterest{
    public static void main (String[] args){
        Scanner in = new Scanner(System.in); // Create a Scanner object
        System.out.print("Enter Principle Amount: ");
        float Principle = in.nextFloat();
        System.out.print("Enter Interest in %: ");
        float Interest = in.nextFloat();
        Interest += 100;
        System.out.print("Enter Time: ");
        float Time = in.nextFloat();
        float Amount = Interest * Time * Principle / 100;
        System.out.println("Final Amount is " + Amount);
    }
}
```

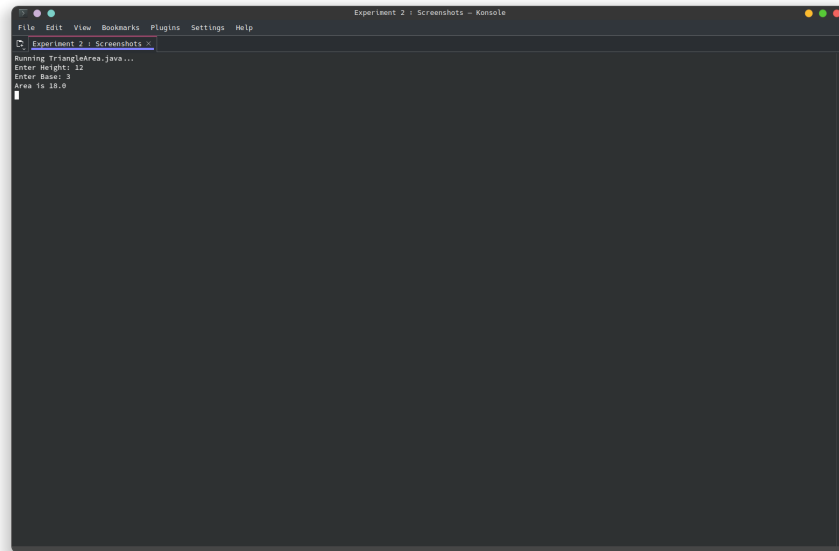


Output

./Experiment 2/TriangleArea.java ##### Code

```
import java.util.Scanner; // Import the Scanner class

class TriangleArea{
    public static void main (String[] args){
        Scanner in = new Scanner(System.in); // Create a Scanner object
        System.out.print("Enter Height: ");
        float Height = in.nextFloat();
        System.out.print("Enter Base: ");
        float Base = in.nextFloat();
        float Area = Height * Base * 1 / 2;
        System.out.println("Area is " + Area);
    }
}
```

Output

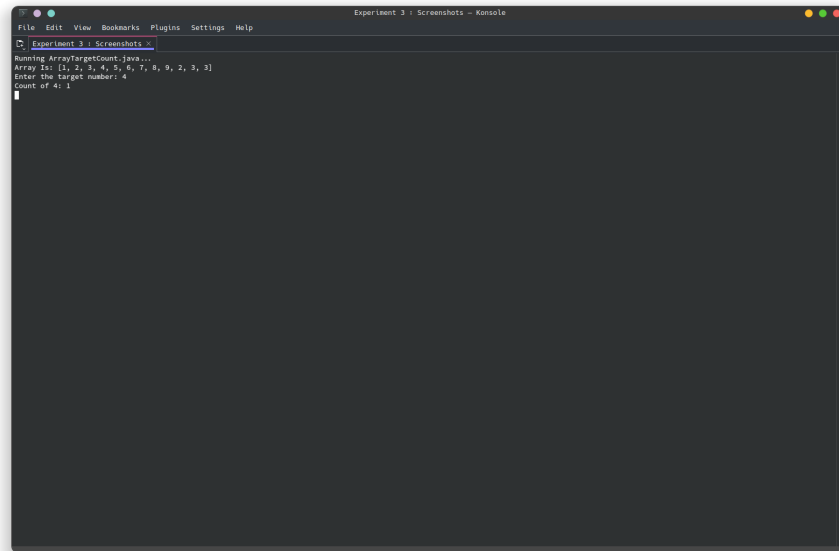
./Experiment 3 ##### ./Experiment 3/ArrayTargetCount.java ##### Code

```
import java.util.Scanner;
import java.util.Arrays;
public class ArrayTargetCount {
    public static void main(String[] args) {
        int[] numbers = {1,2,3,4,5,6,7,8,9, 2, 3,3}; // Example array
        System.out.print("Array Is: ");
        System.out.println(Arrays.toString(numbers));
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the target number: ");
        int target = scanner.nextInt();
        scanner.close();

        int count = 0;
        for (int num : numbers) {
            if (num == target) {
                count++;
            }
        }

        System.out.println("Count of " + target + ": " + count);
    }
}
```



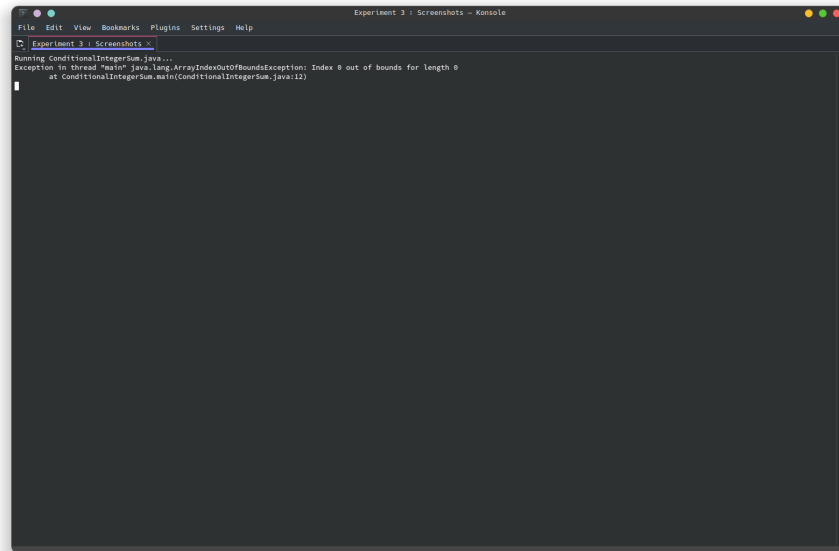
Output

./Experiment 3/ConditionalIntegerSum.java #### Code

```
class ConditionalIntegerSum{
    static int[] dividend = {6, 9};

    public static boolean Divisible(int num) {
        for (int i : dividend) {
            if (num % i == 0) return true;
        }
        return false;
    }

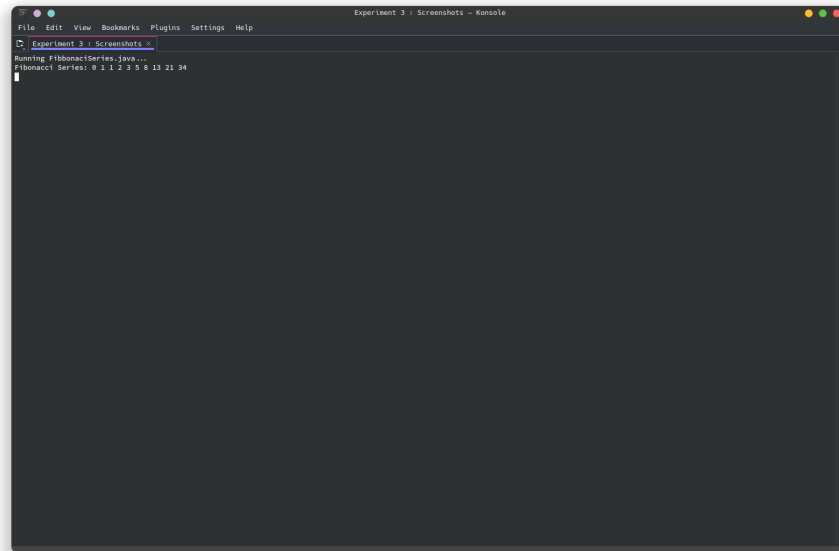
    public static void main (String[] args){
        int startingValue = Integer.parseInt(args[0]);
        int endValue = Integer.parseInt(args[1]);
        int total = 0;
        for (int i = startingValue; i <= endValue ; i++ ) {
            if (Divisible(i)) total += i;
        }
        System.out.print("Total Is: ");
        System.out.println(total);
    }
}
```



Output

./Experiment 3/FibonacciSeries.java ##### Code

```
class FibonacciSeries {  
    public static void main(String[] args) {  
        int n = 10; // Number of terms  
        int first = 0, second = 1;  
  
        System.out.print("Fibonacci Series: " + first + " " + second);  
  
        for (int i = 2; i < n; i++) {  
            int next = first + second;  
            System.out.print(" " + next);  
            first = second;  
            second = next;  
        }  
        System.out.println("");  
    }  
}
```



Output

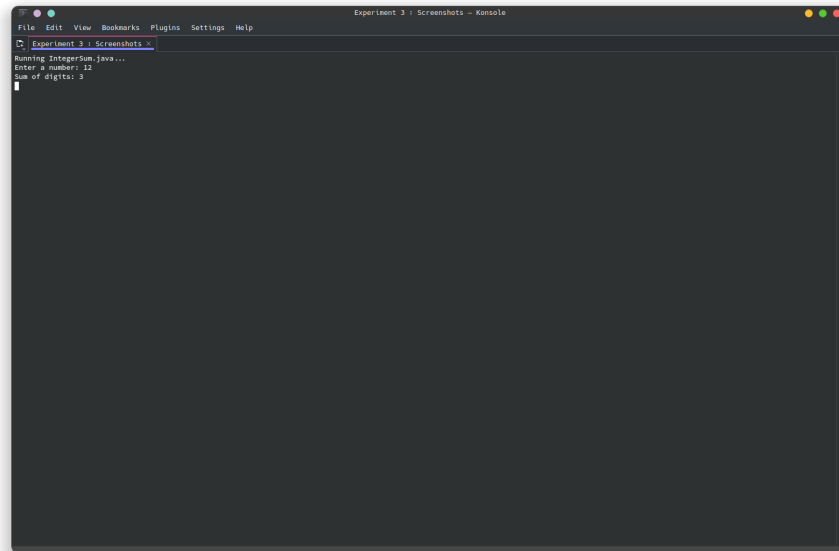
./Experiment 3/IntegerSum.java #### Code

```
import java.util.Scanner;

public class IntegerSum {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int num = scanner.nextInt();
        scanner.close();

        int sum = 0;
        while (num > 0) {
            sum += num % 10;
            num /= 10;
        }

        System.out.println("Sum of digits: " + sum);
    }
}
```



Output

./Experiment 3/Pattern.java ##### Code

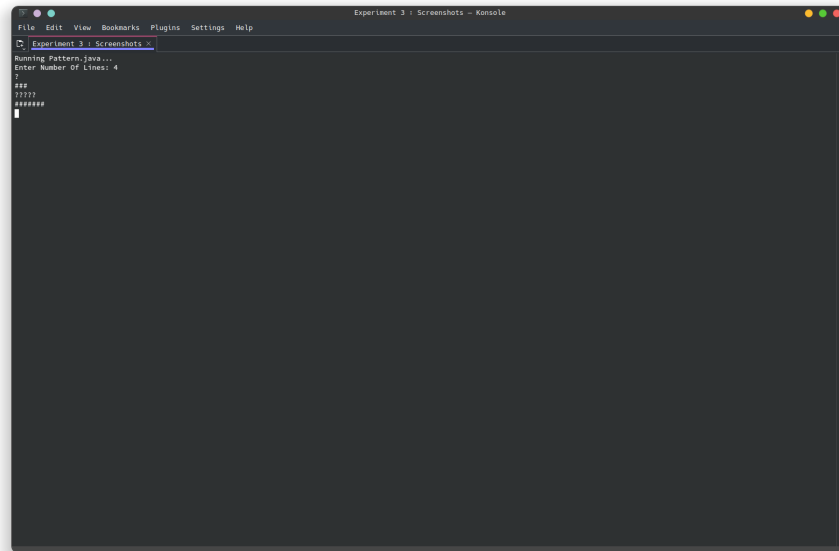
```

/*
?
###
?????
#####
?????????
*/
import java.util.Scanner;

public class Pattern {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter Number Of Lines: ");
        int lines = scanner.nextInt();
        scanner.close();
        String[] str = {"?", "#"};
        for (int i = 0; i < lines; i++) {
            System.out.println(str[i % str.length].repeat(2 * i + 1));
        }
    }
}

```



Output

./Experiment 3/PrimeCount.java ##### Code

```
import java.util.Scanner;

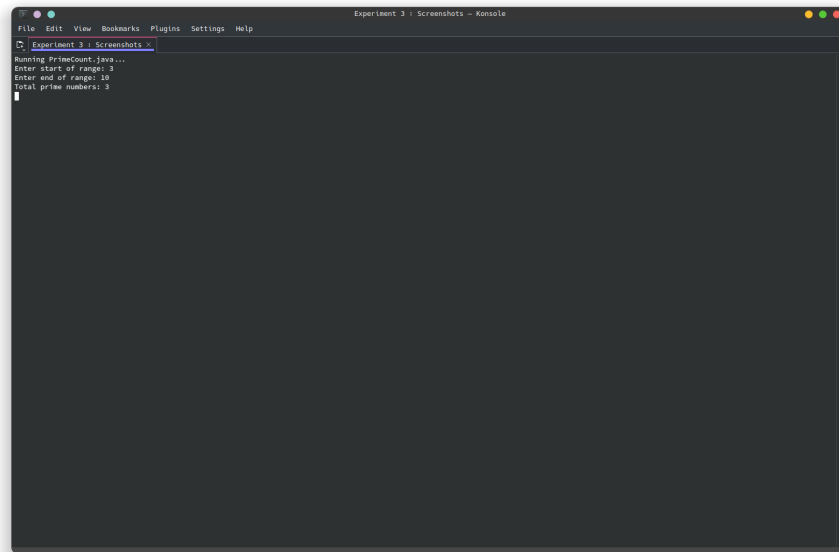
public class PrimeCount {
    static boolean isPrime(int num) {
        if (num < 2) return false;
        for (int i = 2; i * i <= num; i++) {
            if (num % i == 0) return false;
        }
        return true;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter start of range: ");
        int start = scanner.nextInt();
        System.out.print("Enter end of range: ");
        int end = scanner.nextInt();
        scanner.close();

        int count = 0;
        for (int i = start; i <= end; i++) {
            if (isPrime(i)) count++;
        }

        System.out.println("Total prime numbers: " + count);
    }
}
```

```
}
}
```



Output

./Experiment 3/SecondLargetElement.java ##### Code

```
public class SecondLargetElement {
    public static void main(String[] args) {
        int[] arr = {10, 20, 5, 8, 25, 22};

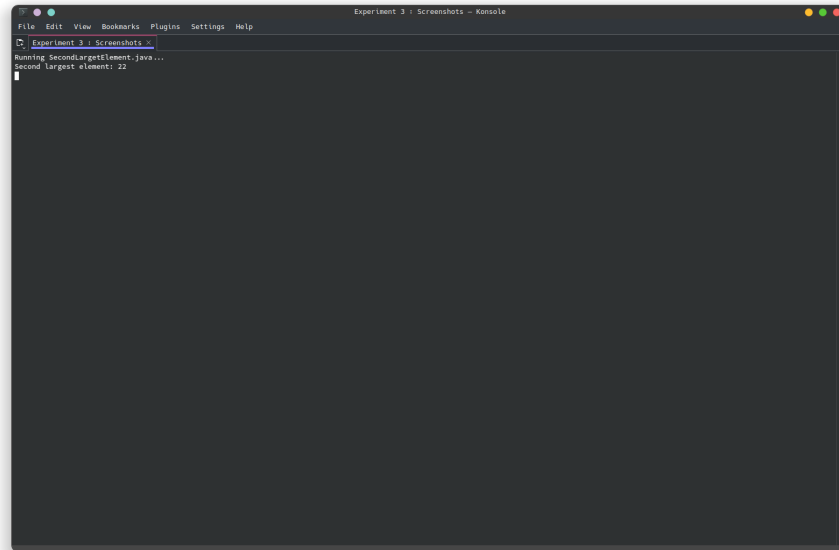
        if (arr.length < 2) {
            System.out.println("Array must have at least two elements.");
            return;
        }

        int first = Integer.MIN_VALUE, second = Integer.MIN_VALUE;

        for (int num : arr) {
            if (num > first) {
                second = first;
                first = num;
            } else if (num > second && num != first) {
                second = num;
            }
        }

        System.out.println("Second largest element: " + (second == Integer.MIN_VALUE ? "No s
    }
}
```

}



Output

./Experiment 4 ##### ./Experiment 4/BankAccount.java ##### Code

*// Create a BankAccount1 class with a private variable balance to store the account balance
// Implement a public method deposit(double amount) to add funds, a protected method
// withdraw(double amount) to deduct funds, and a default-access method checkBalance() to
// display the current balance. Create an object of the class and demonstrate which methods
// and variables can be accessed both inside and outside the class*

```
class BankAccount1 {  
    private double balance;  
    public BankAccount1(double initialBalance) {  
        this.balance = initialBalance;  
    }  
  
    public void deposit(double amount) {  
        if (amount > 0) {  
            balance += amount;  
            System.out.println("Deposited: " + amount);  
        } else {  
            System.out.println("Invalid deposit amount!");  
        }  
    }  
  
    protected void withdraw(double amount) {  
        if (amount > 0 && amount <= balance) {  
            balance -= amount;  
        }  
    }  
}
```



```

        System.out.println("Withdrawn: " + amount);
    } else {
        System.out.println("Insufficient funds or invalid amount!");
    }
}

void checkBalance() {
    System.out.println("Current Balance: " + balance);
}

}

public class BankAccount {
    public static void main(String[] args) {

        BankAccount1 myAccount = new BankAccount1(1000);

        myAccount.deposit(500);

        myAccount.checkBalance();

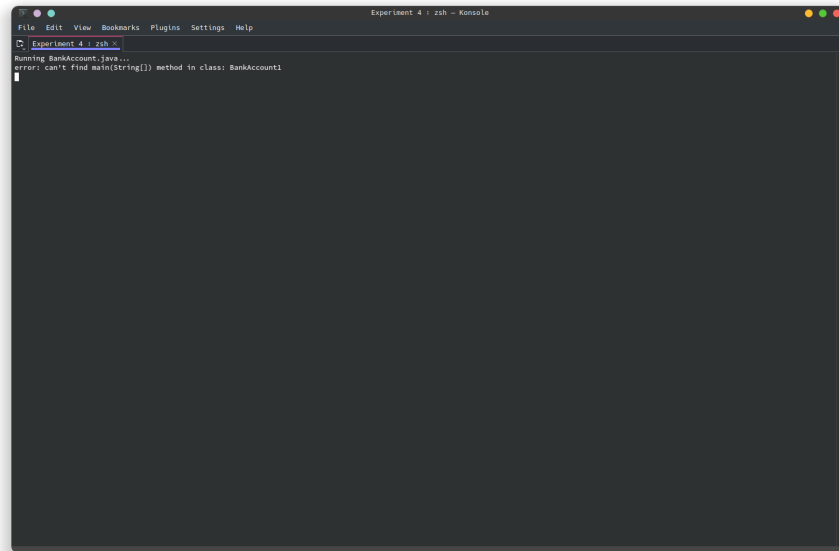
        myAccount.withdraw(300);

        myAccount.checkBalance();

        // System.out.println(myAccount.balance); // No

    }
}

```



Output

./Experiment 4/Calculator.java ##### Code

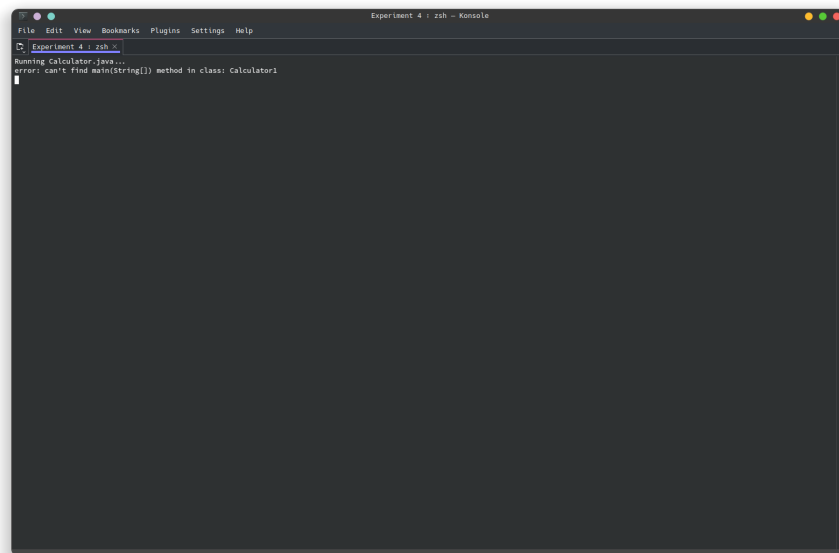
```
// Create a Calculator class that contains a method add() to perform addition. Overload the  
// add() method to handle different types and numbers of parameters, such as adding two  
// integers, two doubles, and three integers. Create an object of the class and demonstrate  
// method variations.  
// Define the Calculator class  
class Calculator1 {  
  
    // Method 1: Add two integers  
    public int add(int a, int b) {  
        return a + b;  
    }  
  
    // Method 2: Add two doubles  
    public double add(double a, double b) {  
        return a + b;  
    }  
  
    // Method 3: Add three integers  
    public int add(int a, int b, int c) {  
        return a + b + c;  
    }  
}  
  
// Main class to test the Calculator  
public class Calculator {
```

```

    public static void main(String[] args) {
        // Create an object of Calculator
        Calculator1 calc = new Calculator1();

        // Demonstrating overloaded methods
        System.out.println("Sum of two integers: " + calc.add(5, 10));
        System.out.println("Sum of two doubles: " + calc.add(5.5, 10.2));
        System.out.println("Sum of three integers: " + calc.add(5, 10, 15));
    }
}

```



Output

./Experiment 4/Company.java ##### Code

```

// A company wants to develop an Employee Management System to track employee details
// such as name, department, salary, and employee ID. The system should also calculate the
// total salary expenditure and keep a record of the total number of employees. Implement a
// Java program by creating an Employee class that includes instance variables for employee
// ID, name, department, and salary. The class should have a default constructor that
// initializes employee details with default values and a parameterized constructor that sets
// employee details based on user input. Use a static variable totalEmployees to track the
// number of employees and implement a static method to display this count. Additionally,
// define a method calculateSalary() that returns the salary of the employee and another
// method displayEmployeeInfo() to display all employee details. To ensure data
// encapsulation, mark the salary variable as private and provide a public method to access
// Declare the totalEmployees variable as static so that it is shared among all instances.
// main method, create multiple Employee objects using both default and parameterized
// constructors. Use the this keyword in the constructors to distinguish between class

```

```

// variables and constructor parameters. Finally, display the total number of employees and
// the salary details for each employee. The program should successfully demonstrate the
// behavior of static and non-static members, the initialization of objects using construct
// and the role of access modifiers in an employee management scenari

```

```

// Main Company Class

```

```

public class Company {
    static final int Employee_Count = 10;
    static private Employee[] Employees = new Employee[Employee_Count];

```

```

// Main

```

```

public static void main(String[] args) {
    Employees[0] = new Employee();
    for (int i = 1; i < Employees.length ; i++) {
        Employees[i] = new Employee("Name", "Department", 10);
    }
    System.out.println("Total Number Of Employees: " + Employee.TotalEmployees());
    TotalSalary();
    for (int i = 1; i < Employees.length ; i++) {
        Employees[i].DisplayEmployeeInfo();
    }
}

```

```

// Total Salary

```

```

public static double TotalSalary(){
    double total = 0;
    for (Employee Emp : Employees) {
        total += Emp.calculateSalary();
    }
    System.out.println("Total Expenditure: " + total);
    return total;
}

```

```

// Employee Class

```

```

class Employee{
    static int totalEmployees;
    int ID;
    String name;
    String department;
    private double salary;

```

```

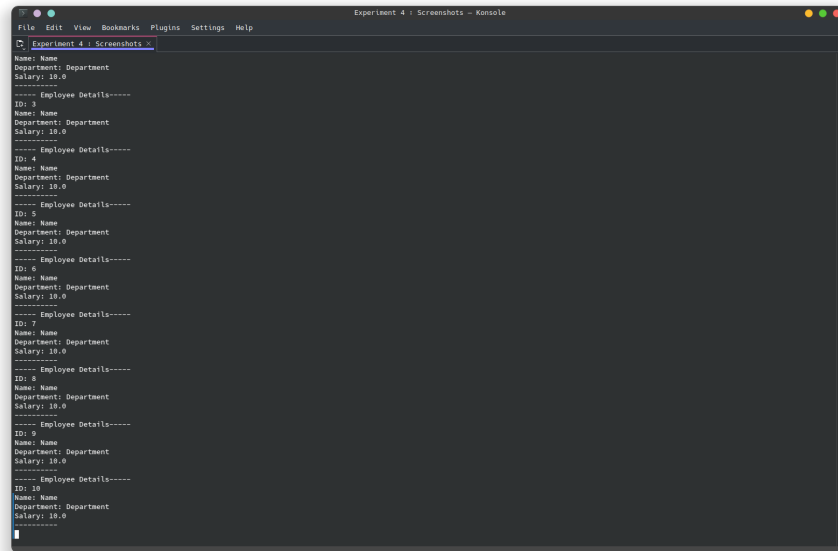
Employee(){
    this.ID = totalEmployees + 1;
    this.name = "Default Name";
    this.department = "Default Department";
    this.salary = 0;
    totalEmployees += 1;
}
Employee(String name, String department, double salary){
    this.ID = totalEmployees + 1;
    this.name = name;
    this.department = department;
    this.salary = salary;
    totalEmployees += 1;
}

public double calculateSalary(){
    // System.out.println("Salary for ID " + ID + " is " + this.salary);
    return salary;
}

public static int TotalEmployees(){
    return totalEmployees;
}

public void DisplayEmployeeInfo(){
    System.out.println("----- Employee Details-----");
    System.out.println("ID: " + ID);
    System.out.println("Name: " + name);
    System.out.println("Department: " + department);
    System.out.println("Salary: " + salary);
    System.out.println("-----");
}
}

```



Output

./Experiment 4/CourseRegistration.java ##### Code

*// A student is developing a course registration system that allows students to enroll in
 // courses. Each course has a course name and a course code. Implement a Course class with
 // appropriate attributes and use the "this" keyword to differentiate between class attribut
 // and constructor parameters during initialization. Create an object of the Course class an
 // display the course details*

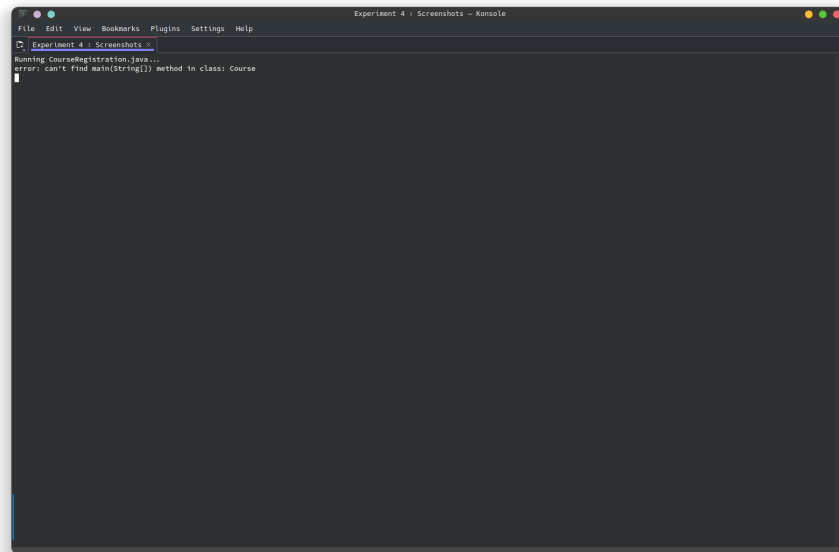
```
class Course{
    public String CourseName;
    public String CourseCode;

    public Course(String CourseName, String CourseCode){
        this.CourseName = CourseName;
        this.CourseCode = CourseCode;
    }

    public void displayDetails(){
        System.out.println("Course Name: " + this.CourseName);
        System.out.println("Course Code: " + this.CourseCode);
    }
}

class CourseRegistration{
    public static void main(String[] args) {
        Course Maths = new Course("Maths", "Math100");
        Course CompSci = new Course("CompSci", "CompSci100");
        Maths.displayDetails();
        CompSci.displayDetails();
    }
}
```

```
}  
}
```



Output

./Experiment 4/Student.java ##### Code

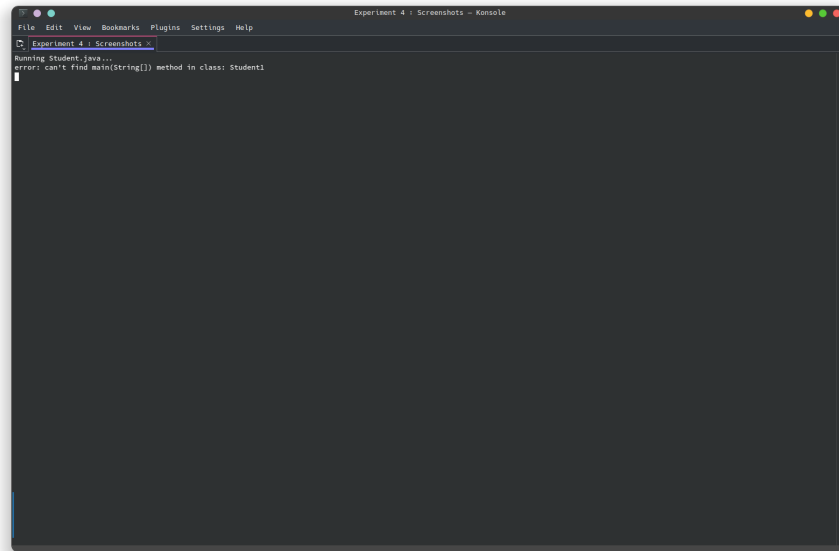
*// Create a Student class with attributes for name and age. Implement a default constructor
// assign default values and a parameterized constructor to initialize the attributes with
// defined values. Create objects using both constructors and display their details.*

```
class Student1 {  
    String Name;  
    int Age;  
    Student1(String Name, int Age){  
        this.Name = Name;  
        this.Age = Age;  
        System.out.print("Name: " + this.Name + "\n" + ("Age: " + this.Age + "\n"));  
    }  
    Student1(){  
        this.Name = "Default Name";  
        this.Age = 0;  
        System.out.print("Name: " + this.Name + "\n" + ("Age: " + this.Age + "\n"));  
    }  
}  
  
public class Student{  
    public static void main(String[] args) {  
        System.out.println("Default Student");  
        Student1 DefaultStudent1 = new Student1();  
        System.out.println("Named Student");  
    }  
}
```

```

        Student1 NamedStudent1 = new Student1("Named Student1", 12);
    }
}

```



Output

./Experiment 4/University.java ##### Code

```

// Create a Student class that has a static variable universityName and a non-static variable
// studentName. Include a static method to display the university name. Then, create multiple
// student objects to demonstrate how the static variable is shared among all instances, while
// the non-static variable holds unique values for each object
class Student {
    static String universityName = "Super Cool University"; // Static
    String studentName; // Non Static

    public Student(String studentName) {
        this.studentName = studentName;
    }

    static void displayUniversity() {
        System.out.println("University: " + universityName);
    }

    void displayStudent() {
        System.out.println("Student: " + studentName + "\nUniversity: " + universityName);
    }
}

```



```

public class University {
    public static void main(String[] args) {

        System.out.println("----- Initial Name -----");
        Student.displayUniversity();

        Student student1 = new Student("Student 1");
        Student student2 = new Student("Student 2");

        student1.displayStudent();
        student2.displayStudent();

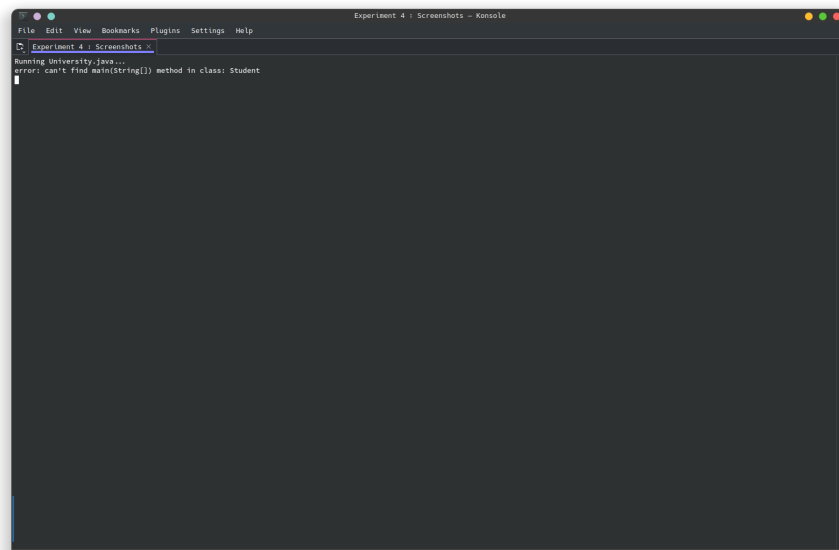
        System.out.println("----- Changed Name -----");

        Student.universityName = "University 2";

        Student.displayUniversity();

        student1.displayStudent();
        student2.displayStudent();
    }
}

```



Output

./Experiment 5 ### ./Experiment 5/EmployeeSalary.java #### Code

```

class Employee {
    protected String name;
    protected int empid;
    protected double salary;

    public Employee() {
        this.name = "Default Name";
        this.empid = 0;
        this.salary = 0.0;
    }

    public Employee(String name, int empid, double salary) {
        this.name = name;
        this.empid = empid;
        this.salary = salary;
    }

    public String getName() {
        System.out.println("Name is " + name);
        return name;
    }

    public double getSalary() {
        System.out.println("Salary of " + name + " is " + salary);
        return salary;
    }

    public void increaseSalary(double percentage) {
        salary += salary * (percentage / 100);
        System.out.println("New Salary: " + salary);
    }
}

class Manager extends Employee {
    private String department;

    public Manager(String name, int empid, double salary, String department) {
        super(name, empid, salary);
        this.department = department;
    }

    public String getDepartment() {
        return department;
    }

    @Override

```

```

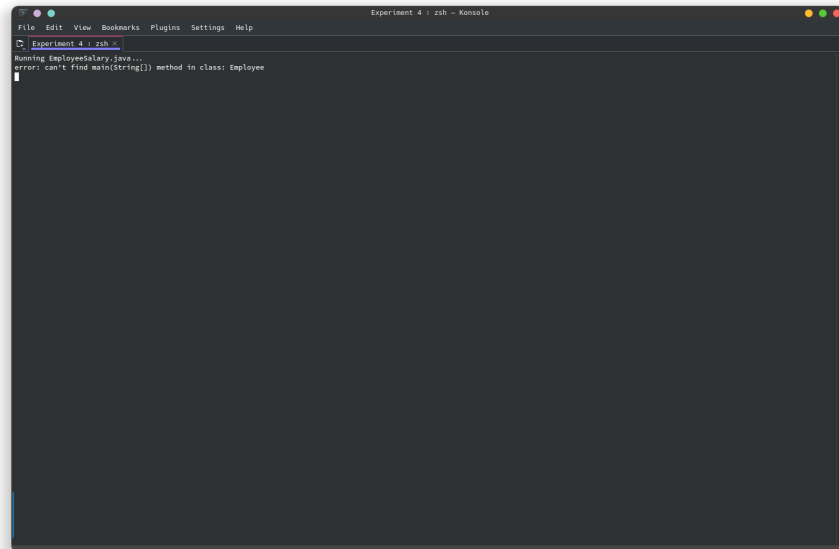
        public String getName() {
            System.out.println(name + " (Manager of " + department + ")");
            return name + " (Manager of " + department + ")";
        }
    }

    public class EmployeeSalary {
        public static void main(String[] args) {
            Employee emp1 = new Employee("Caterpillar", 101, 50000);
            emp1.getName();
            emp1.getSalary();
            emp1.increaseSalary(10);

            System.out.println("-----");

            Manager emp2 = new Manager("Butterfly", 201, 70000, "IT");
            emp2.getName();
            emp2.getSalary();
            emp2.increaseSalary(15);
        }
    }
}

```



Output

./Experiment 5/PlayerSubclasses.java #### Code

```

class Player {
    protected String name;
    protected int age;
}

```

```

    protected String position;

    public Player(String name, int age, String position) {
        this.name = name;
        this.age = age;
        this.position = position;
    }

    public void play() {
        System.out.println(name + " is playing as a " + position);
    }

    public void train() {
        System.out.println("training" + name);
    }
}

class Cricket_Player extends Player {
    public Cricket_Player(String name, int age, String position) {
        super(name, age, position);
    }

    @Override
    public void play() {
        System.out.println(name + " is playing cricket as a " + position);
    }

    @Override
    public void train() {
        System.out.println(name + " is practicing cricket.");
    }
}

class Football_Player extends Player {
    public Football_Player(String name, int age, String position) {
        super(name, age, position);
    }

    @Override
    public void play() {
        System.out.println(name + " is playing football as a " + position);
    }

    @Override
    public void train() {
        System.out.println(name + " is practicing football.");
    }
}

```

```

    }
}

class Hockey_Player extends Player {
    public Hockey_Player(String name, int age, String position) {
        super(name, age, position);
    }

    @Override
    public void play() {
        System.out.println(name + " is playing hockey as a " + position);
    }

    @Override
    public void train() {
        System.out.println(name + " is practicing Hockey.");
    }
}

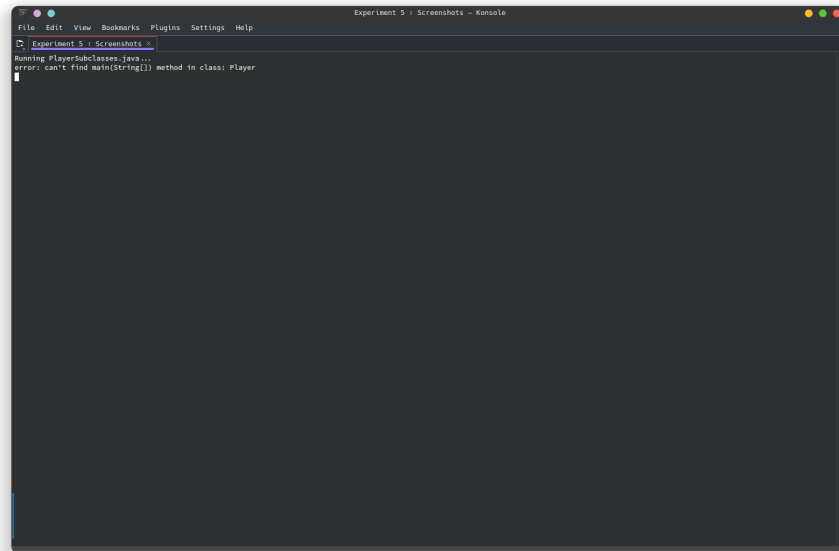
public class PlayerSubclasses {
    public static void main(String[] args) {
        Cricket_Player cricketer = new Cricket_Player("Virat Kohli", 35, "Batsman");
        Football_Player footballer = new Football_Player("Lionel Messi", 37, "Forward");
        Hockey_Player hockeyPlayer = new Hockey_Player("Manpreet Singh", 32, "Midfielder");

        System.out.println("Cricket Player");
        cricketer.play();
        cricketer.train();

        System.out.println("Football Player");
        footballer.play();
        footballer.train();

        System.out.println("Hockey Player");
        hockeyPlayer.play();
        hockeyPlayer.train();
    }
}

```



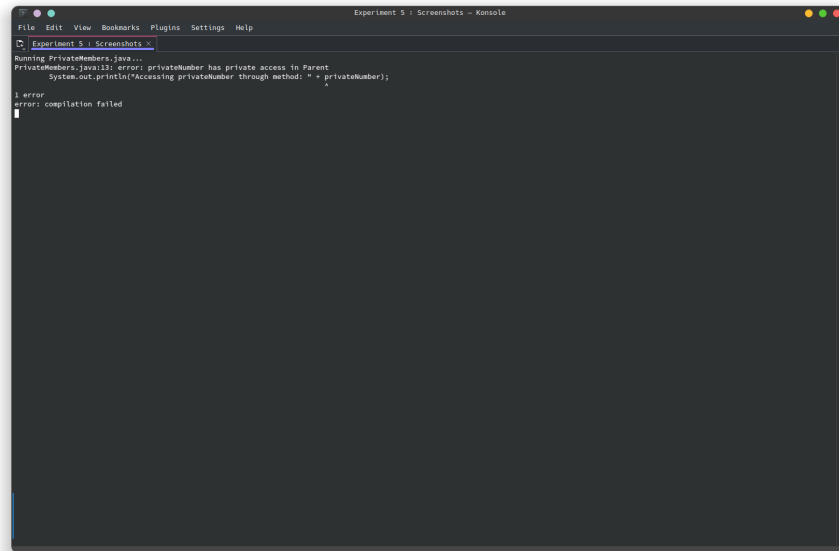
Output

./Experiment 5/PrivateMembers.java ##### Code

```
// Write a Java program to demonstrate that a private member of a superclass cannot be accessed
class Parent {
    private int privateNumber = 42;
    public int getPrivateNumber() {
        return privateNumber;
    }
}

class Child extends Parent {
    public void display() {
        System.out.println("Accessing privateNumber through method: " + getPrivateNumber());
    }
    public void access() {
        System.out.println("Accessing privateNumber through method: " + privateNumber);
    }
}

public class PrivateMembers {
    public static void main(String[] args) {
        Child obj = new Child();
        obj.display();
        // obj.access(); // Access Error
    }
}
```



Output

./Experiment 5/TrunkCalls.java #### Code

```
import java.util.Scanner;

class TrunkCall {
    protected double rate;

    public double computeCharge(int duration) {
        return duration * rate;
    }
}

class OrdinaryCall extends TrunkCall {
    public OrdinaryCall() {
        this.rate = 1.5;
    }
}

class UrgentCall extends TrunkCall {
    public UrgentCall() {
        this.rate = 2.5;
    }
}

class LightningCall extends TrunkCall {
    public LightningCall() {
        this.rate = 4.0;
    }
}
```

```

    }
}

public class TrunkCalls {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter call duration (in minutes): ");
        int duration = scanner.nextInt();

        System.out.println("Select Call Type:");
        System.out.println("1. Ordinary");
        System.out.println("2. Urgent");
        System.out.println("3. Lightning");
        System.out.print("Enter choice (1-3): ");
        int choice = scanner.nextInt();

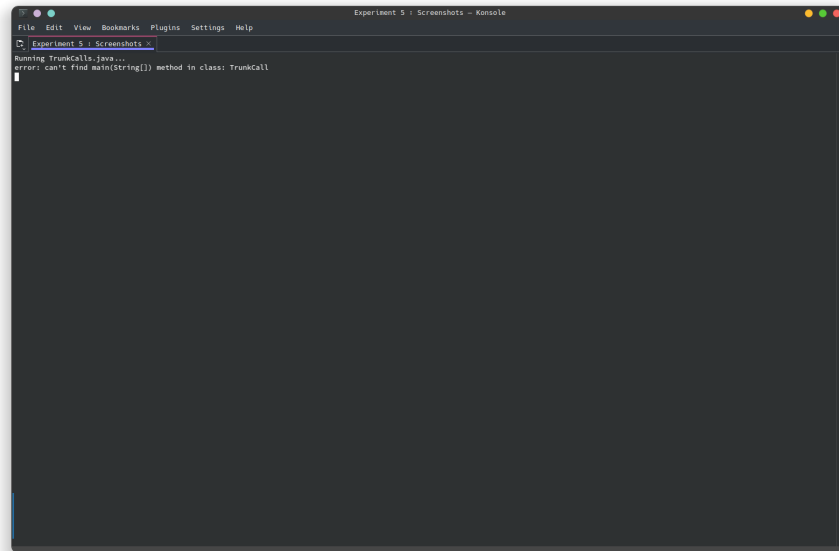
        TrunkCall call;

        switch (choice) {
            case 1:
                call = new OrdinaryCall();
                break;
            case 2:
                call = new UrgentCall();
                break;
            case 3:
                call = new LightningCall();
                break;
            default:
                System.out.println("Invalid choice! Defaulting to Ordinary call.");
                call = new OrdinaryCall();
        }

        double totalCharge = call.computeCharge(duration);
        System.out.println("Total Trunk Call Charge: $" + totalCharge);

        scanner.close();
    }
}

```

Output

./Experiment 5/University.java ##### Code

```
class Person {  
    private String name;  
    private int age;  
    private String address;  
  
    public Person(String name, int age, String address) {  
        this.name = name;  
        this.age = age;  
        this.address = address;  
    }  
  
    public void displayDetails() {  
        System.out.println("Name: " + name);  
        System.out.println("Age: " + age);  
        System.out.println("Address: " + address);  
    }  
}
```

```
class Staff extends Person {  
    private String staffId;  
    private String department;  
  
    public Staff(String name, int age, String address, String staffId, String department) {  
        super(name, age, address);  
    }  
}
```

```

        this.staffId = staffId;
        this.department = department;
    }

    @Override
    public void displayDetails() {
        super.displayDetails();
        System.out.println("Staff ID: " + staffId);
        System.out.println("Department: " + department);
    }
}

class Professor extends Staff {
    private String specialization;

    public Professor(String name, int age, String address, String staffId, String department) {
        super(name, age, address, staffId, department);
        this.specialization = specialization;
    }

    @Override
    public void displayDetails() {
        super.displayDetails();
        System.out.println("Specialization: " + specialization);
    }

    public void conductLecture() {
        System.out.println("Professor " + getName() + " is conducting a lecture on " + specialization);
    }

    private String getName() {
        return "Professor";
    }
}

class Student extends Person {
    private String studentId;
    private String course;

    public Student(String name, int age, String address, String studentId, String course) {
        super(name, age, address);
        this.studentId = studentId;
        this.course = course;
    }

    @Override

```

```

        public void displayDetails() {
            super.displayDetails();
            System.out.println("Student ID: " + studentId);
            System.out.println("Course: " + course);
        }
    }

    class GraduateStudent extends Student {
        private String researchTopic;

        public GraduateStudent(String name, int age, String address, String studentId, String course) {
            super(name, age, address, studentId, course);
            this.researchTopic = researchTopic;
        }

        @Override
        public void displayDetails() {
            super.displayDetails();
            System.out.println("Research Topic: " + researchTopic);
        }

        public void submitThesis() {
            System.out.println("Graduate student " + getName() + " has submitted their thesis on " + researchTopic);
        }

        private String getName() {
            return "Graduate Student";
        }
    }

    public class University{
        public static void main(String[] args) {
            Professor professor = new Professor("Dr. Doctor", 45, "Some Address", "S123", "Computer Science");

            GraduateStudent gradStudent = new GraduateStudent("Graduate Idiot", 25, "Somewhere else", "S456", "Biology");

            Person[] people = new Person[2];
            people[0] = professor;
            people[1] = gradStudent;

            for (Person person : people) {
                person.displayDetails();
                System.out.println();
            }

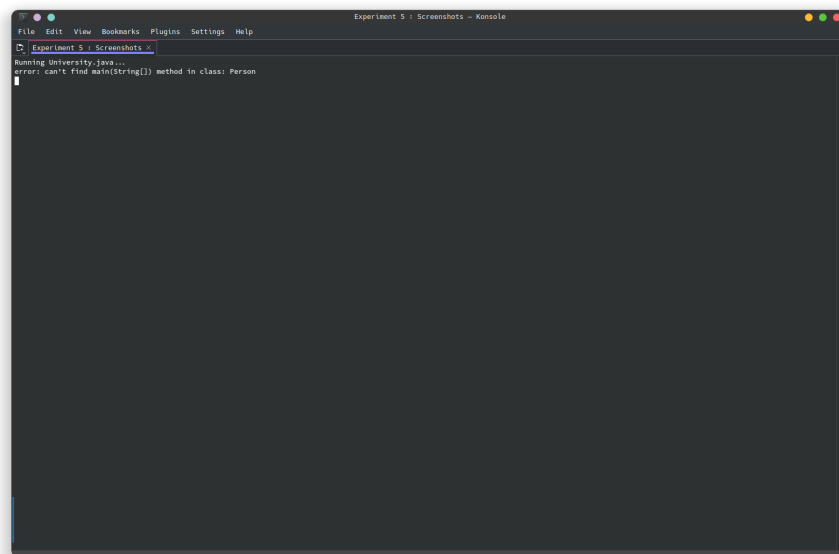
            if (person instanceof Professor) {
                ((Professor) person).conductLecture();
            }
        }
    }

```

```

    } else if (person instanceof GraduateStudent) {
        ((GraduateStudent) person).submitThesis();
    }
    System.out.println("-----");
}
}
}

```



Output

./Experiment 5/VehicleManufacturing.java #### Code

```

class Vehicle {
    private String brand;
    private String model;
    private double price;

    public Vehicle(String brand, String model, double price) {
        this.brand = brand;
        this.model = model;
        this.price = price;
    }

    public void displayDetails() {
        System.out.println("Brand: " + brand);
        System.out.println("Model: " + model);
        System.out.println("Price: " + price);
    }
}

```

```

class Car extends Vehicle {
    private int seatingCapacity;
    private String fuelType;

    public Car(String brand, String model, double price, int seatingCapacity, String fuelType) {
        super(brand, model, price);
        this.seatingCapacity = seatingCapacity;
        this.fuelType = fuelType;
    }

    @Override
    public void displayDetails() {
        super.displayDetails();
        System.out.println("Seating Capacity: " + seatingCapacity);
        System.out.println("Fuel Type: " + fuelType);
    }
}

class ElectricCar extends Car {
    private double batteryCapacity;
    private double chargingTime;

    public ElectricCar(String brand, String model, double price, int seatingCapacity, String fuelType, double batteryCapacity, double chargingTime) {
        super(brand, model, price, seatingCapacity, fuelType);
        this.batteryCapacity = batteryCapacity;
        this.chargingTime = chargingTime;
    }

    @Override
    public void displayDetails() {
        super.displayDetails();
        System.out.println("Battery Capacity: " + batteryCapacity);
        System.out.println("Charging Time: " + chargingTime);
    }
}

class Motorcycle extends Vehicle {
    private double engineCapacity;
    private String type;

    public Motorcycle(String brand, String model, double price, double engineCapacity, String type) {
        super(brand, model, price);
        this.engineCapacity = engineCapacity;
        this.type = type;
    }
}

```

```

        @Override
        public void displayDetails() {
            super.displayDetails();
            System.out.println("Engine Capacity: " + engineCapacity);
            System.out.println("Type: " + type);
        }
    }

    public class VehicleManufacturing {
        public static void main(String[] args) {
            Vehicle vehicle = new Vehicle("Generic", "V1", 10000);
            Car car = new Car("Toyota", "Corolla", 20000, 5, "Petrol");

            ElectricCar electricCar = new ElectricCar("Tesla", "Model S", 80000, 5, "Electric",

            Motorcycle motorcycle = new Motorcycle("Harley-Davidson", "Sportster", 15000, 1200,

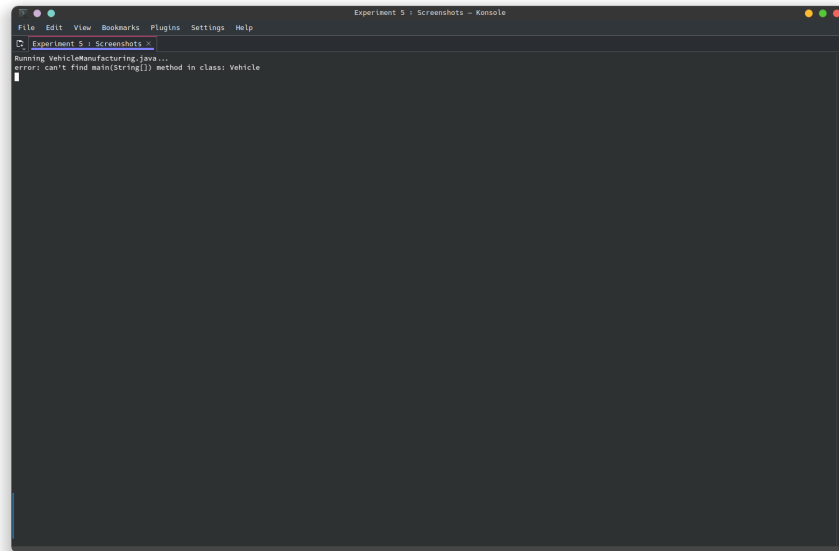
            System.out.println("Vehicle Details:");
            vehicle.displayDetails();
            System.out.println("-----");

            System.out.println("Car Details:");
            car.displayDetails();
            System.out.println("-----");

            System.out.println("Electric Car Details:");
            electricCar.displayDetails();
            System.out.println("-----");

            System.out.println("Motorcycle Details:");
            motorcycle.displayDetails();
            System.out.println("-----");
        }
    }
}

```



Output

./Experiment 5/WorkerSubclasses.java ##### Code

```
//
// A company wants to develop an Employee Management System to track employee details
// such as name, department, salary, and employee ID. The system should also calculate the
// total salary expenditure and keep a record of the total number of employees. Implement a
// Java program by creating an Employee class that includes instance variables for employee
// ID, name, department, and salary. The class should have a default constructor that
// initializes employee details with default values and a parameterized constructor that sets
// employee details based on user input. Use a static variable totalEmployees to track the
// number of employees and implement a static method to display this count. Additionally,
// define a method calculateSalary() that returns the salary of the employee and another
// method displayEmployeeInfo() to display all employee details. To ensure data
// encapsulation, mark the salary variable as private and provide a public method to access
// Declare the totalEmployees variable as static so that it is shared among all instances.
// main method, create multiple Employee objects using both default and parameterized
// constructors. Use the this keyword in the constructors to distinguish between class
// variables and constructor parameters. Finally, display the total number of employees and
// the salary details for each employee. The program should successfully demonstrate the
// behavior of static and non-static members, the initialization of objects using constructors
// and the role of access modifiers in an employee management scenario.
class Worker {
    protected String name;
    protected double salaryRate;

    public Worker(String name, double salaryRate) {
        this.name = name;
```

```

        this.salaryRate = salaryRate;
    }

    public double computePay(int hours) {
        return salaryRate * hours;
    }
}

class DailyWorker extends Worker {
    public DailyWorker(String name, double salaryRate) {
        super(name, salaryRate);
    }

    @Override
    public double computePay(int daysWorked) {
        return daysWorked * 8 * salaryRate;
    }
}

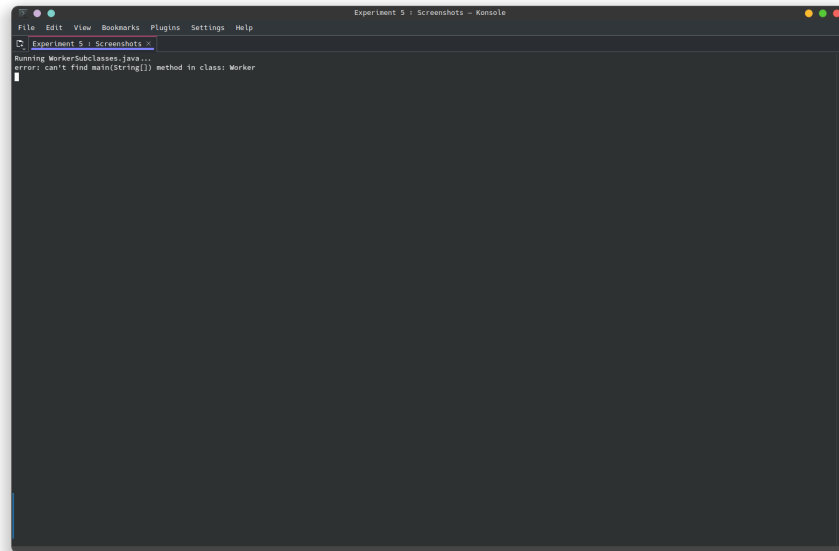
class SalariedWorker extends Worker {
    public SalariedWorker(String name, double salaryRate) {
        super(name, salaryRate);
    }

    @Override
    public double computePay(int hours) {
        return 40 * salaryRate;
    }
}

public class WorkerSubclasses {
    public static void main(String[] args) {
        Worker dailyWorker = new DailyWorker("Daily Worker", 15.5);
        Worker salariedWorker = new SalariedWorker("Salaried Worker", 20.0);

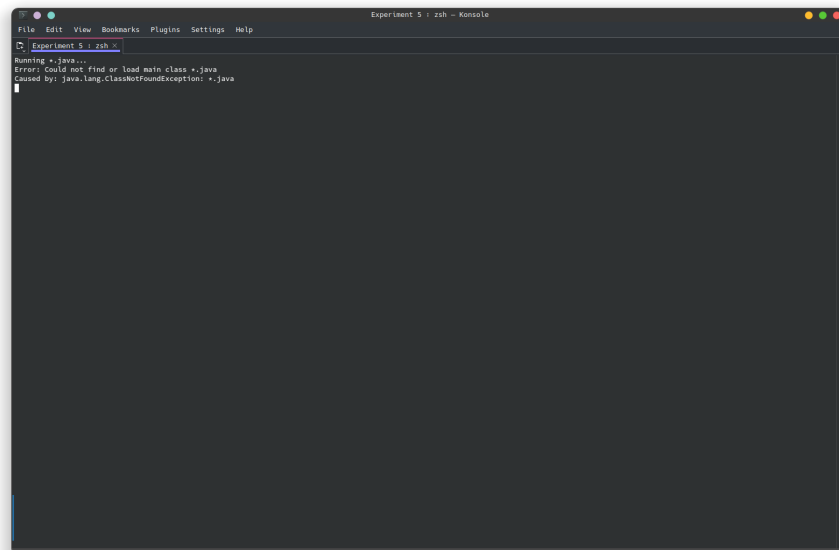
        System.out.println(dailyWorker.name + "'s Weekly Pay: " + dailyWorker.computePay(5));
        System.out.println(salariedWorker.name + "'s Weekly Pay: " + salariedWorker.computePay(5));
    }
}

```

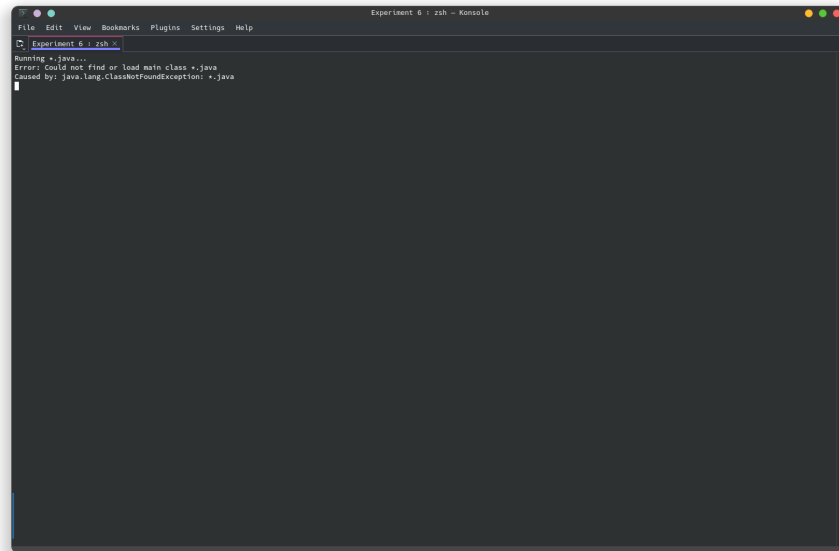
Output

./Experiment 6 ### ./Experiment 6/*.java ##### Code



Output

./Experiment 7 ### ./Experiment 7/*.java ##### Code



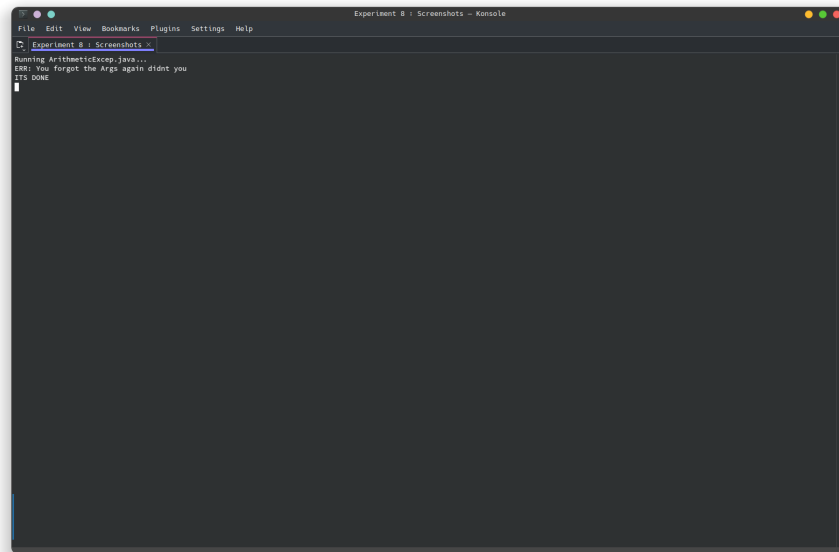
Output

./Experiment 8 ### ./Experiment 8/ArithmeticExcep.java ##### Code

```
// Write a Java program that takes two integers as input from the user and  
// performs division.  
// Handle the ArithmeticException that occurs if the denominator is zero. Use a  
// try-catch block  
// to catch the exception and display an appropriate error message.  
// Additionally, use a finally  
// block to print "Operation completed" regardless of whether an exception  
// occurs or not.
```

```
class ArithmeticExcep {  
    public static void main(String[] args) {  
        try {  
  
            if (Integer.parseInt(args[1]) == 0) {  
                throw new ArithmeticException("Division by zero!");  
            }  
  
            System.out.println(args[0] + "/" + args[1] + "=" + Float.parseFloat(args[0]) / Float.parseFloat(args[1]));  
        } catch (ArrayIndexOutOfBoundsException e) {  
            System.out.println("ERR: You forgot the Args again didnt you");  
        } catch (ArithmeticException e) {  
            System.out.println("ERR: Oh crap its an infinity again");  
        } finally {  
            System.out.println("ITS DONE");  
        }  
    }  
}
```

```
}
}
```

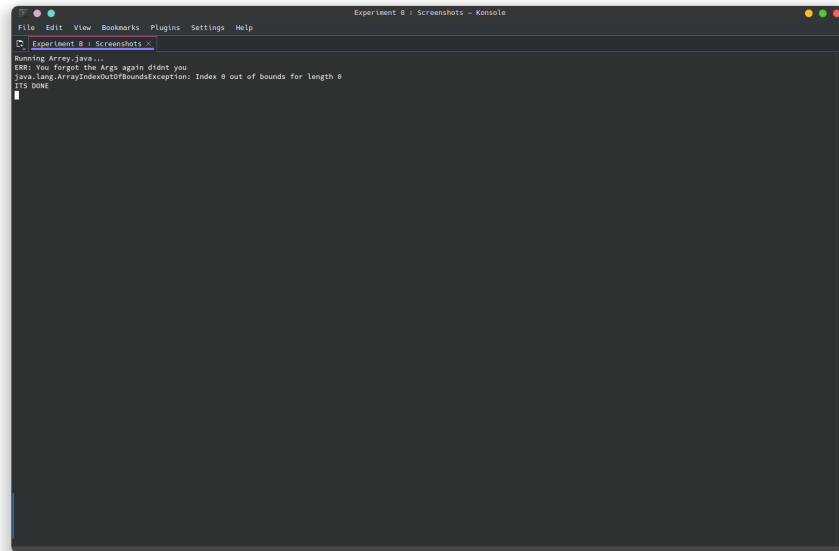


Output

./Experiment 8/Arrey.java ##### Code

```
// Write a Java program that creates an array of 5 integers and asks the user to
// enter an index to
// access the array element. Handle the ArrayIndexOutOfBoundsException if the
// user enters an
// invalid index. Use a try-catch block to catch the exception and display an
// appropriate error
// message. Use the finally block to print "Array access attempted."
```

```
class Arrey {
    public static void main(String[] args) {
        int[] arr = { 2, 3, 5, 7, 11 }; // Array literal
        try {
            System.out.println("Element is: " + arr[Integer.parseInt(args[0])]);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("ERR: You forgot the Args again didnt you\n" + e);
        } finally {
            System.out.println("ITS DONE");
        }
    }
}
```



Output

./Experiment 8/Car.java ##### Code

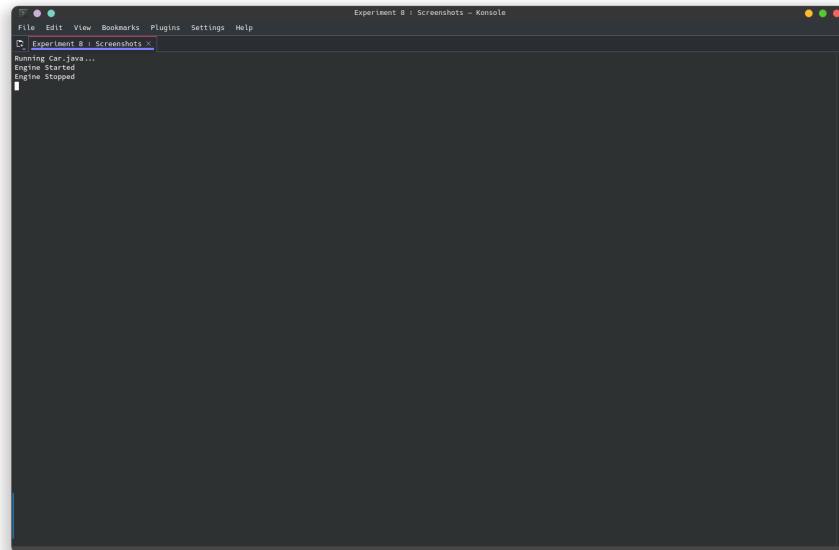
```
// Create an outer class Car with an inner class Engine. The Engine class should  
// have a method  
// start() that prints "Engine started" and a method stop() that prints "Engine  
// stopped". The Car  
// class should have a method drive() that creates an instance of the Engine  
// class and calls its  
// start() and stop() methods
```

```
class Car {  
    static void drive() {  
        Engine e = new Engine();  
        e.Start();  
        e.Stop();  
    }  
  
    static class Engine {  
        void Start() {  
            System.out.println("Engine Started");  
        }  
  
        void Stop() {  
            System.out.println("Engine Stopped");  
        }  
    }  
}  
  
public static void main(String[] args) {
```

```

        drive();
    }
}

```



Output

./Experiment 8/Event.java ##### Code

```

// Create an interface EventHandler with a method handleEvent(). In the main
// method, demonstrate the use of:
// a. A local inner class inside a method registerEvent() that implements
// EventHandler and prints "Event handled by local inner class".
// b. An anonymous inner class that implements EventHandler and prints "Event
// handled by anonymous inner class".

```

```

interface EventHandler {
    public void handleEvent();
}

class Event {

    public static void registerEvent() {
        class LocalInner implements EventHandler {
            @Override
            public void handleEvent() {
                System.out.println("Handled by Local Inner Class");
            }
        }
    }
}

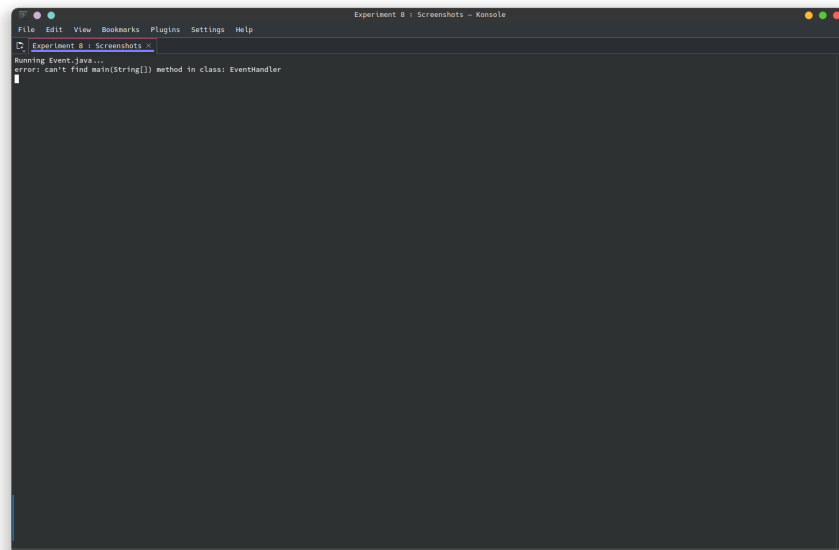
```

```

        LocalInner l = new LocalInner();
        l.handleEvent();
    }

    public static void main(String[] args) {
        // A
        registerEvent();
        // B
        EventHandler anonymousHandler = new EventHandler() {
            @Override
            public void handleEvent() {
                System.out.println("Event handled by anonymous inner class");
            }
        };
        anonymousHandler.handleEvent();
    }
}

```



Output

./Experiment 8/FileName.java #### Code

```

// Write a Java program that reads a file name from the user and attempts to
// open and read the
// file. Define a method readFile() that throws a FileNotFoundException using
// the throws
// keyword. In the main method, call this method and handle the exception using
// a try-catch

```

```

// block. Display an appropriate message if the file is not found. Use a finally
// block to ensure a
// message like "File operation attempted" is printed.

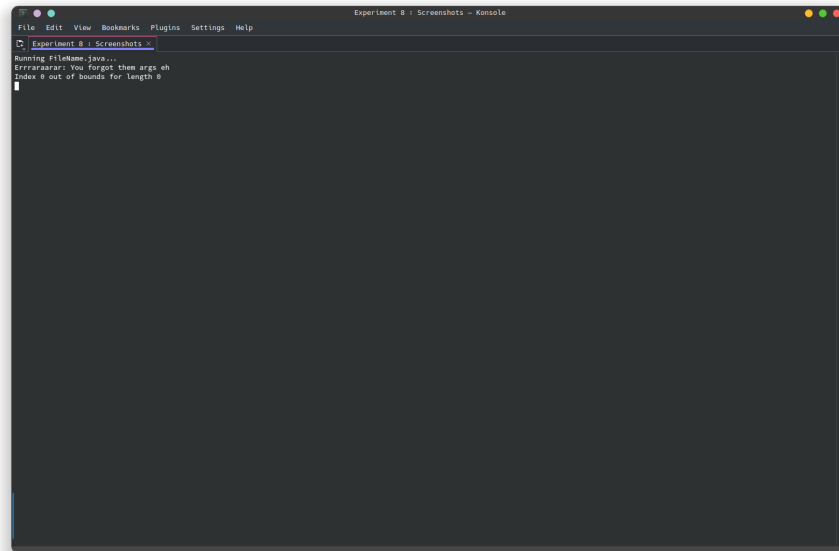
import java.io.File;
import java.io.FileNotFoundException;

class FileName {

    public static Boolean checkFileExists(String filePath) throws FileNotFoundException {
        File file = new File(filePath);
        if (!file.exists()) {
            throw new FileNotFoundException("File not found: " + filePath);
        } else {
            return true;
        }
    }

    public static void main(String[] args) {
        try {
            System.out.println(checkFileExists(args[0]) ? "YAY FOUND IT!!!! IT EXISTS CONGRATULATIONS"
                : "nothing matters it doesnt exist");
        } catch (IndexOutOfBoundsException e) {
            System.err.println("Errrraraarar: You forgot them args eh \n" + e.getMessage());
        } catch (FileNotFoundException e) {
            System.err.println("Erawr: " + e.getMessage());
        }
    }
}

```



Output

./Experiment 8/FileRead.java #### Code

```
// Write a Java program that reads the contents of a file named student.txt  
// using FileReader and  
// displays the data on the console. Handle FileNotFoundException if the file  
// does not exist and  
// display an appropriate error message. Use a try-catch block for exception  
// handling.
```

```
import java.io.*;
```

```
class FileRead {  
    public static void main(String[] args) throws IOException {
```

```
        // Reading File name
```

```
        String path;
```

```
        try {
```

```
            path = args[0];
```

```
            FileReader fr = new FileReader(path);
```

```
            int i;
```

```
            while ((i = fr.read()) != -1)
```

```
                System.out.print((char) i);
```

```
        } catch (IndexOutOfBoundsException e) {
```

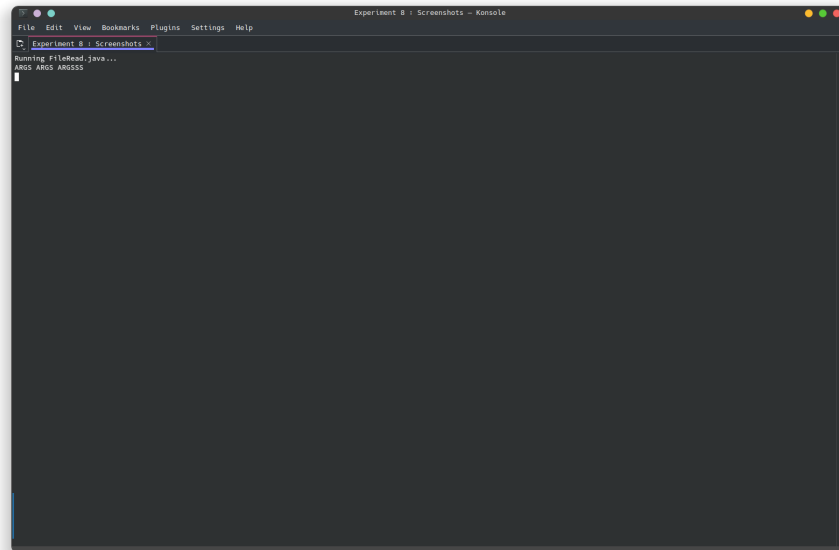
```
            System.out.println("ARGS ARGS ARGSSS");
```



```

    }
}
}

```



Output

./Experiment 8/FileWrite.java ##### Code

```

// Write a Java program that takes user input for a student's name, roll number,
// and grade, and
// writes this information to a file named student.txt using FileWriter. Ensure
// the program
// appends the data to the file if it already exists. Handle any exceptions
// using try-catch and
// display an appropriate message if an error occurs.
// Sample File Content:
// Name: Aman, Roll Number: 120112, Grade: A
// Name: Parul, Roll Number: 120131, Grade: B
import java.io.File;
import java.io.FileWriter;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.BufferedWriter;
import java.io.FileInputStream;
import java.util.Scanner;

class FileWrite {
    public static void fileAppend(String filename, String[] args) throws IOException {

```

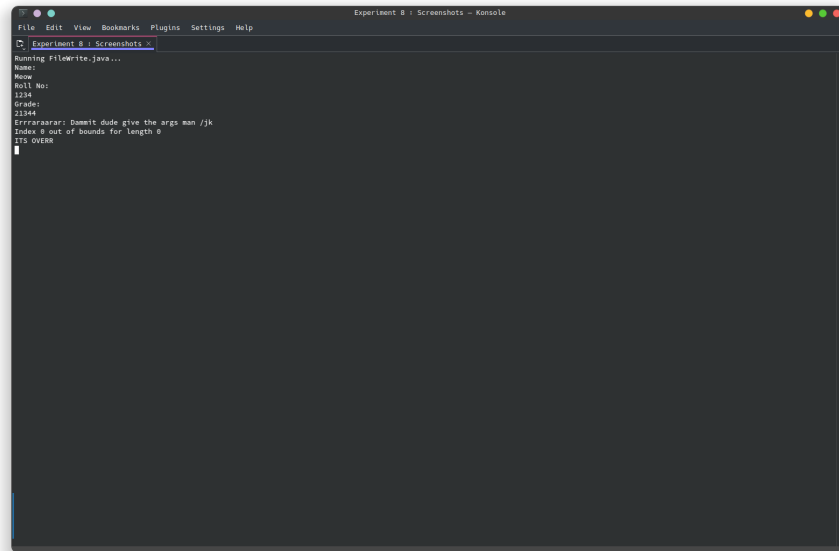
```

        FileWriter fw = new FileWriter(filename, true);
        BufferedWriter bw = new BufferedWriter(fw);
        bw.write("Name: " + args[0] + " Roll No: " + args[1] + " Grade: " + args[2]);
        bw.newLine();
        bw.close();
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Name:");
        String num1 = scanner.nextLine();
        System.out.println("Roll No:");
        String num2 = scanner.nextLine();
        System.out.println("Grade:");
        String num3 = scanner.nextLine();
        String s[] = { num1, num2, num3 };
        try {
            fileAppend(args[0], s);
        } catch (IndexOutOfBoundsException e) {
            System.err.println("Errraraarar: Dammit dude give the args man /jk \n" + e.getMessage());
        } catch (IOException e) {
            System.err.println(e.getMessage());
        } finally {
            scanner.close();
            System.out.println("ITS OVERR");
        }
    }
}

```



Output

./Experiment 8/Library.java ##### Code

// Create an outer class Library with a static nested class Book. The Book class should have attributes like title, author, and ISBN, and a method displayDetails() to print these details. In the main method, create an instance of the Book class and call displayDetails() to show book information.

```

class Library {
    static class Book {
        String Author;
        String Title;
        String ISBN;

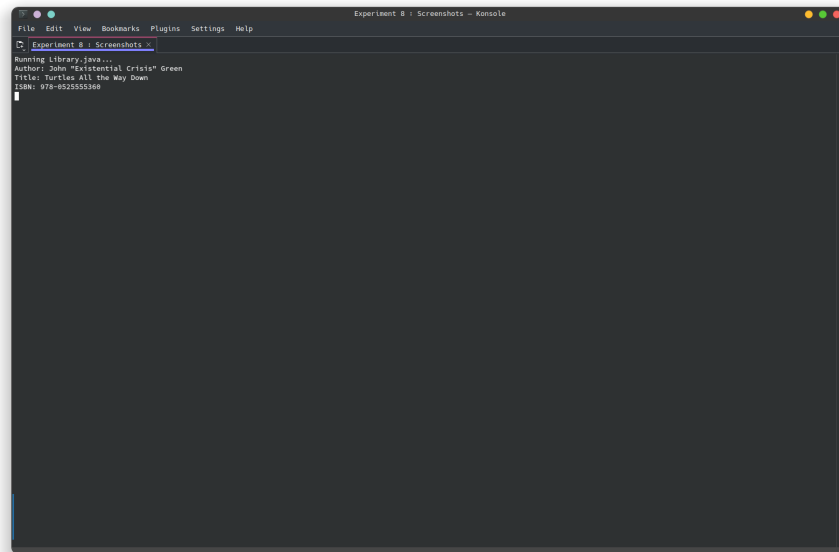
        Book(String Author, String Title, String ISBN) {
            this.Author = Author;
            this.Title = Title;
            this.ISBN = ISBN;
        }

        public void displayDetails() {
            System.out.println("Author: " + Author + "\nTitle: " + Title + "\nISBN: " + ISBN);
        }
    }

    public static void main(String[] args) {
        Book b = new Book("John \"Existential Crisis\" Green", "Turtles All the Way Down", "978-0-13-035812-5");
        b.displayDetails();
    }
}

```

}



Output

```
## ./Experiment 9 ### ./Experiment 9/Calculator.java ##### Code
```

```
// Java program to create a simple calculator
// with basic +, -, /, * using java swing elements

import java.awt.event.*;
import javax.swing.*;
import java.awt.*;

class calculator extends JFrame implements ActionListener {
    // create a frame
    static JFrame f;

    // create a textfield
    static JTextField l;

    // store operator and operands
    String s0, s1, s2;

    calculator() {
        s0 = s1 = s2 = "";
    }

    public static void main(String args[]) {
```

```

f = new JFrame("calculator");

try {
    UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
} catch (Exception e) {
    System.err.println(e.getMessage());
}

calculator c = new calculator();

l = new JTextField(16);

l.setEditable(false);

// create number buttons and some operators
JButton b0, b1, b2, b3, b4, b5, b6, b7, b8, b9, ba, bs, bd, bm, be, beq, beq1;

// create number buttons
b0 = new JButton("0");
b1 = new JButton("1");
b2 = new JButton("2");
b3 = new JButton("3");
b4 = new JButton("4");
b5 = new JButton("5");
b6 = new JButton("6");
b7 = new JButton("7");
b8 = new JButton("8");
b9 = new JButton("9");

// equals button
beq1 = new JButton("=");

// create operator buttons
ba = new JButton("+");
bs = new JButton("-");
bd = new JButton("/");
bm = new JButton("*");
beq = new JButton("C");

// create . button
be = new JButton(".");

// create a panel
JPanel p = new JPanel();

// add action listeners

```

```

bm.addActionListener(c);
bd.addActionListener(c);
bs.addActionListener(c);
ba.addActionListener(c);
b9.addActionListener(c);
b8.addActionListener(c);
b7.addActionListener(c);
b6.addActionListener(c);
b5.addActionListener(c);
b4.addActionListener(c);
b3.addActionListener(c);
b2.addActionListener(c);
b1.addActionListener(c);
b0.addActionListener(c);
be.addActionListener(c);
beq.addActionListener(c);
beq1.addActionListener(c);

// add elements to panel
p.add(1);
p.add(ba);
p.add(b1);
p.add(b2);
p.add(b3);
p.add(bs);
p.add(b4);
p.add(b5);
p.add(b6);
p.add(bm);
p.add(b7);
p.add(b8);
p.add(b9);
p.add(bd);
p.add(be);
p.add(b0);
p.add(beq);
p.add(beq1);

// set Background of panel
p.setBackground(Color.blue);

// add panel to frame
f.add(p);

f.setSize(200, 220);
f.show();

```

```

}

public void actionPerformed(ActionEvent e) {
    String s = e.getActionCommand();

    // if the value is a number
    if ((s.charAt(0) >= '0' && s.charAt(0) <= '9') || s.charAt(0) == '.') {
        // if operand is present then add to second no
        if (!s1.equals(""))
            s2 = s2 + s;
        else
            s0 = s0 + s;

        // set the value of text
        l.setText(s0 + s1 + s2);
    } else if (s.charAt(0) == 'C') {
        // clear the one letter
        s0 = s1 = s2 = "";

        // set the value of text
        l.setText(s0 + s1 + s2);
    } else if (s.charAt(0) == '=') {

        double te;

        // store the value in 1st
        if (s1.equals("+"))
            te = (Double.parseDouble(s0) + Double.parseDouble(s2));
        else if (s1.equals("-"))
            te = (Double.parseDouble(s0) - Double.parseDouble(s2));
        else if (s1.equals("/"))
            te = (Double.parseDouble(s0) / Double.parseDouble(s2));
        else
            te = (Double.parseDouble(s0) * Double.parseDouble(s2));

        // set the value of text
        l.setText(s0 + s1 + s2 + "=" + te);

        // convert it to string
        s0 = Double.toString(te);

        s1 = s2 = "";
    } else {
        // if there was no operand
        if (s1.equals("") || s2.equals(""))
            s1 = s;
    }
}

```

```

// else evaluate
else {
    double te;

    // store the value in 1st
    if (s1.equals("+"))
        te = (Double.parseDouble(s0) + Double.parseDouble(s2));
    else if (s1.equals("-"))
        te = (Double.parseDouble(s0) - Double.parseDouble(s2));
    else if (s1.equals("/"))
        te = (Double.parseDouble(s0) / Double.parseDouble(s2));
    else
        te = (Double.parseDouble(s0) * Double.parseDouble(s2));

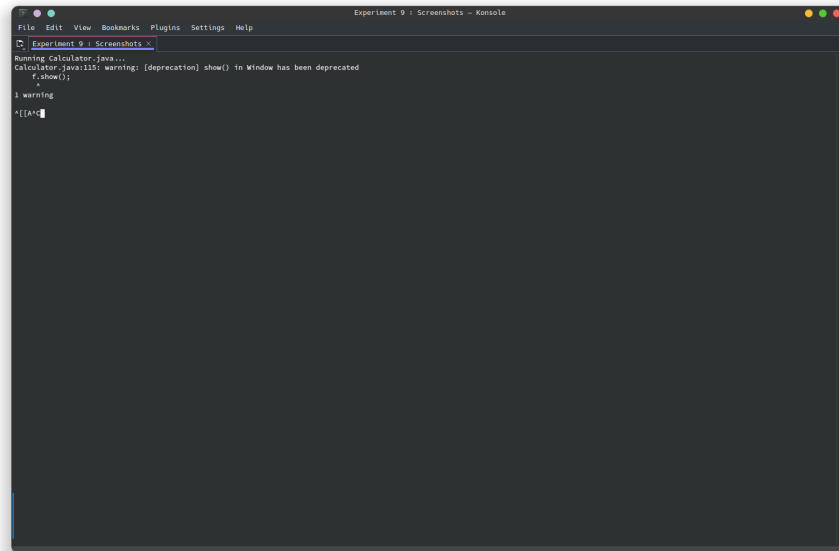
    // convert it to string
    s0 = Double.toString(te);

    // place the operator
    s1 = s;

    // make the operand blank
    s2 = "";
}

// set the value of text
l.setText(s0 + s1 + s2);
}
}
}

```

Output

./Experiment 9/LoginDemo.java ##### Code

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

class LoginDemo extends JFrame implements ActionListener {
    JPanel panel;
    JLabel user_label, password_label, message;
    JTextField userName_text;
    JPasswordField password_text;
    JButton submit, cancel;

    LoginDemo() {
        user_label = new JLabel();
        user_label.setText("User Name :");
        userName_text = new JTextField();
        password_label = new JLabel();
        password_label.setText("Password :");
        password_text = new JPasswordField();
        submit = new JButton("SUBMIT");
        panel = new JPanel(new GridLayout(3, 1));
        panel.add(user_label);
        panel.add(userName_text);
        panel.add(password_label);
        panel.add(password_text);
        message = new JLabel();
    }
}
```

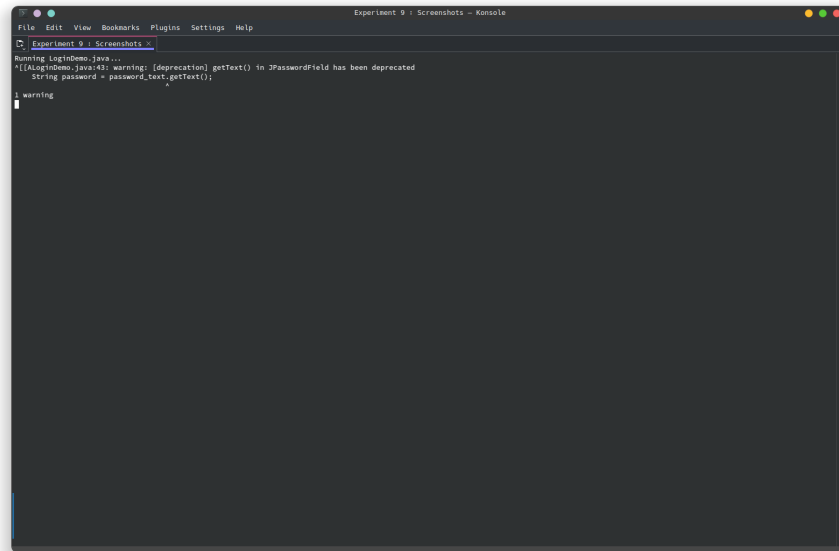
```

        panel.add(message);
        panel.add(submit);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        submit.addActionListener(this);
        add(panel, BorderLayout.CENTER);
        setTitle("Please Login Here !");
        setSize(450, 350);
        setVisible(true);
    }

    public static void main(String[] args) {
        new LoginDemo();
    }

    @Override
    public void actionPerformed(ActionEvent ae) {
        String userName = userName_text.getText();
        String password = password_text.getText();
        if (userName.trim().equals("admin") && password.trim().equals("admin")) {
            message.setText(" Hello " + userName + "");
        } else {
            message.setText(" WRONG WRONG PASSWORD SHAME ON YOU.. ");
        }
    }
}

```



Output

./Experiment 9/ToDoList.java ##### Code

```
import java.awt.Dimension;
import java.awt.Font;
import java.awt.GridLayout;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;

import javax.swing.BorderFactory;
import javax.swing.Box;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.border.Border;
import java.awt.Color;
import java.awt.BorderLayout;
import java.awt.Component;

class Task extends JPanel {

    JLabel index;
    JTextField taskName;
    JButton done;

    Color pink = new Color(255, 161, 161);
```

```

Color green = new Color(188, 226, 158);
Color doneColor = new Color(233, 119, 119);

private boolean checked;

Task() {
    this.setPreferredSize(new Dimension(400, 20)); // set size of task
    this.setBackground(pink); // set background color of task

    this.setLayout(new BorderLayout()); // set layout of task

    checked = false;

    index = new JLabel(""); // create index label
    index.setPreferredSize(new Dimension(20, 20)); // set size of index label
    index.setHorizontalAlignment(JLabel.CENTER); // set alignment of index label
    this.add(index, BorderLayout.WEST); // add index label to task

    taskName = new JTextField("Write something.."); // create task name text field
    taskName.setBorder(BorderFactory.createEmptyBorder()); // remove border of text field
    taskName.setBackground(pink); // set background color of text field

    this.add(taskName, BorderLayout.CENTER);

    done = new JButton("Done");
    done.setPreferredSize(new Dimension(80, 20));
    done.setBorder(BorderFactory.createEmptyBorder());
    done.setBackground(doneColor);
    done.setFocusPainted(false);

    this.add(done, BorderLayout.EAST);
}

public void changeIndex(int num) {
    this.index.setText(num + ""); // num to String
    this.revalidate(); // refresh
}

public JButton getDone() {
    return done;
}

public boolean getState() {
    return checked;
}

```

```

    public void changeState() {
        this.setBackground(green);
        taskName.setBackground(green);
        checked = true;
        revalidate();
    }
}

class List extends JPanel {

    Color lightColor = new Color(252, 221, 176);

    List() {

        GridLayout layout = new GridLayout(10, 1);
        layout.setVgap(5); // Vertical gap

        this.setLayout(layout); // 10 tasks
        this.setPreferredSize(new Dimension(400, 560));
        this.setBackground(lightColor);
    }

    public void updateNumbers() {
        Component[] listItems = this.getComponents();

        for (int i = 0; i < listItems.length; i++) {
            if (listItems[i] instanceof Task) {
                ((Task) listItems[i]).changeIndex(i + 1);
            }
        }
    }

    public void removeCompletedTasks() {

        for (Component c : getComponents()) {
            if (c instanceof Task) {
                if (((Task) c).getState()) {
                    remove(c); // remove the component
                    updateNumbers(); // update the indexing of all items
                }
            }
        }
    }
}

```

```

}

class Footer extends JPanel {

    JButton addTask;
    JButton clear;

    Color orange = new Color(233, 133, 128);
    Color lightColor = new Color(252, 221, 176);
    Border emptyBorder = BorderFactory.createEmptyBorder();

    Footer() {
        this.setPreferredSize(new Dimension(400, 60));
        this.setBackground(lightColor);

        addTask = new JButton("Add Task"); // add task button
        addTask.setBorder(emptyBorder); // remove border
        addTask.setFont(new Font("Sans-serif", Font.ITALIC, 20)); // set font
        addTask.setVerticalAlignment(JButton.BOTTOM); // align text to bottom
        addTask.setBackground(orange); // set background color
        this.add(addTask); // add to footer

        this.add(Box.createHorizontalStrut(20)); // Space between buttons

        clear = new JButton("Clear finished tasks"); // clear button
        clear.setFont(new Font("Sans-serif", Font.ITALIC, 20)); // set font
        clear.setBorder(emptyBorder); // remove border
        clear.setBackground(orange); // set background color
        this.add(clear); // add to footer
    }

    public JButton getNewTask() {
        return addTask;
    }

    public JButton getClear() {
        return clear;
    }
}

class TitleBar extends JPanel {

    Color lightColor = new Color(252, 221, 176);

    TitleBar() {
        this.setPreferredSize(new Dimension(400, 80)); // Size of the title bar
    }
}

```

```

        this.setBackground(lightColor); // Color of the title bar
        JLabel titleText = new JLabel("To Do List"); // Text of the title bar
        titleText.setPreferredSize(new Dimension(200, 60)); // Size of the text
        titleText.setFont(new Font("Sans-serif", Font.BOLD, 20)); // Font of the text
        titleText.setHorizontalAlignment(JLabel.CENTER); // Align the text to the center
        this.add(titleText); // Add the text to the title bar
    }
}

class AppFrame extends JFrame {

    private TitleBar title;
    private Footer footer;
    private List list;

    private JButton newTask;
    private JButton clear;

    AppFrame() {
        this.setSize(400, 600); // 400 width and 600 height
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // Close on exit
        this.setVisible(true); // Make visible

        title = new TitleBar();
        footer = new Footer();
        list = new List();

        this.add(title, BorderLayout.NORTH); // Add title bar on top of the screen
        this.add(footer, BorderLayout.SOUTH); // Add footer on bottom of the screen
        this.add(list, BorderLayout.CENTER); // Add list in middle of footer and title

        newTask = footer.getNewTask();
        clear = footer.getClear();

        addListeners();
    }

    public void addListeners() {
        newTask.addMouseListener(new MouseAdapter() {
            @Override
            public void mousePressed(MouseEvent e) {
                Task task = new Task();
                list.add(task); // Add new task to list
                list.updateNumbers(); // Updates the numbers of the tasks

                task.getDone().addMouseListener(new MouseAdapter() {

```

```

        @Override
        public void mousePressed(MouseEvent e) {

            task.changeState(); // Change color of task
            list.updateNumbers(); // Updates the numbers of the tasks
            revalidate(); // Updates the frame

        }
    });
}

});

clear.addMouseListener(new MouseAdapter() {
    @Override
    public void mousePressed(MouseEvent e) {
        list.removeCompletedTasks(); // Removes all tasks that are done
        repaint(); // Repaints the list
    }
});
}

}

public class ToDoList {
    public static void main(String args[]) {
        AppFrame frame = new AppFrame(); // Create the frame
    }
}

@interface override {

}

```

Output

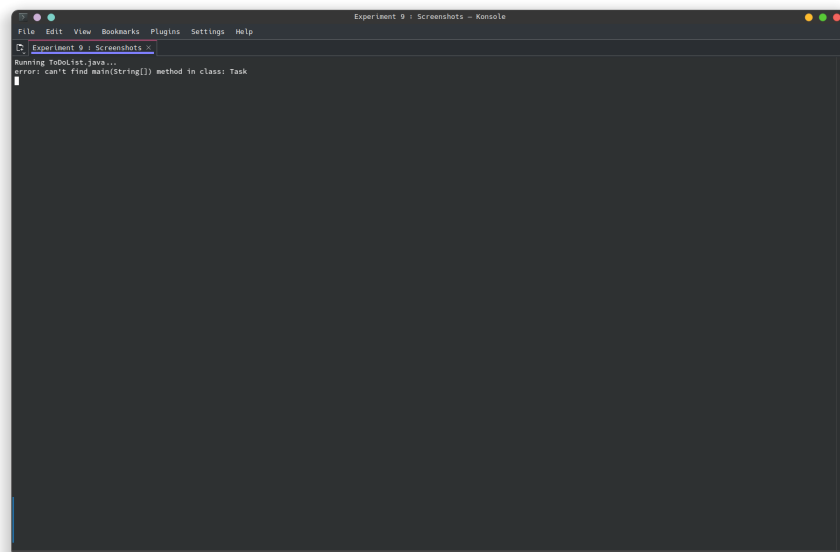


Figure 1: ./Experiment 9/ToDoList.java