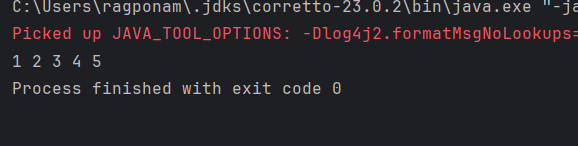
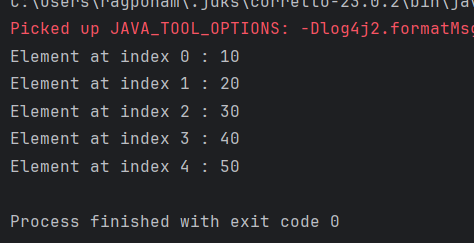
Task021:

public class Task021 {  
 public static void main(String[] args)  
 {  
 int[] arr = { 1, 2, 3, 4, 5 };  
 int n = arr.length;  
 for (int i = 0; i < n; i++)  
 System.*out*.print(arr[i] + " ");  
 }  
}



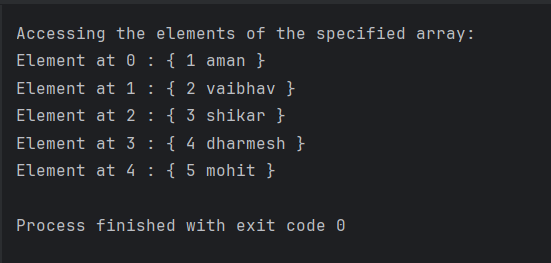
Task022:

class GFG {  
 public static void main(String[] args)  
 {  
 int[] arr;  
 arr = new int[5];  
   
 arr[0] = 10;  
 arr[1] = 20;  
 arr[2] = 30;  
 arr[3] = 40;  
 arr[4] = 50;  
  
 for (int i = 0; i < arr.length; i++)  
 System.*out*.println("Element at index "  
 + i + " : " + arr[i]);  
 }  
}



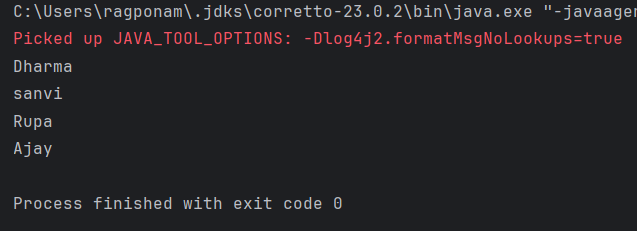
Task023:

class Student {  
 public int roll\_no;  
 public String name;  
  
 Student(int Roll\_no, String Name) {  
 this.roll\_no = Roll\_no;  
 this.name = Name;  
 System.*out*.println("Student object created: " + this.name + " (" + this.roll\_no + ")");  
 }  
}  
  
public class Task023 {  
 public static void main(String[] args) {  
  
 Student[] arr;  
  
 arr = new Student[5];  
  
 System.*out*.println("Initializing Student array:");  
 arr[0] = new Student(1, "aman");  
 arr[1] = new Student(2, "vaibhav");  
 arr[2] = new Student(3, "shikar");  
 arr[3] = new Student(4, "dharmesh");  
 arr[4] = new Student(5, "mohit");  
  
 System.*out*.println("\nAccessing the elements of the specified array:");  
 for (int i = 0; i < arr.length; i++) {  
 System.*out*.println("Element at " + i + " : { "  
 + arr[i].roll\_no + " "  
 + arr[i].name + " }");  
 }  
 }  
}



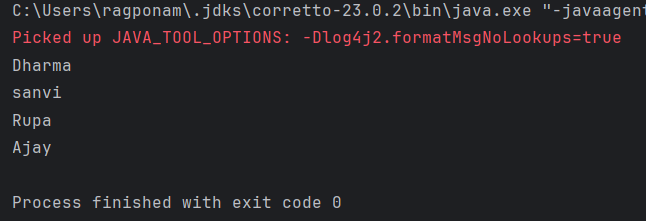
Task024:

class Student{  
 public String name;  
 Student(String name){  
 this.name = name;  
 }  
  
 @Override  
 public String toString(){  
 return name;  
 }  
}  
  
public class Task024{  
 public static void main (String[] args){  
  
 // declares an Array and initializing the  
 // elements of the array  
 Student[] myStudents = new Student[]{  
 new Student("Dharma"),new Student("sanvi"),  
 new Student("Rupa"),new Student("Ajay")  
 };  
 // accessing the elements of the specified array  
 for(Student m:myStudents){  
 System.*out*.println(m);  
 }  
 }  
}



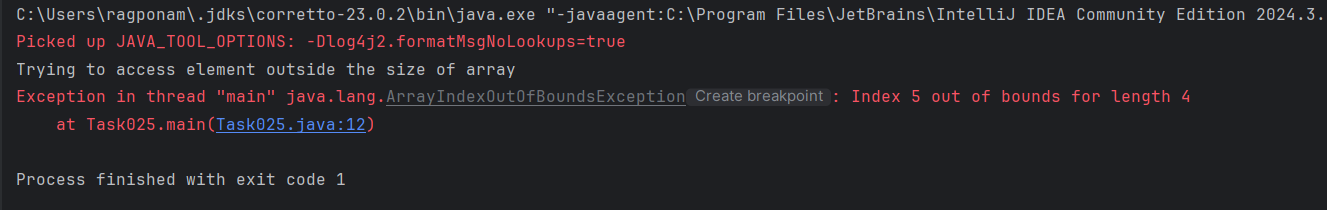
Task024:

class Student{  
 public String name;  
 Student(String name){  
 this.name = name;  
 }  
  
 @Override  
 public String toString(){  
 return name;  
 }  
}  
  
public class Task024{  
 public static void main (String[] args){  
  
 // declares an Array and initializing the  
 // elements of the array  
 Student[] myStudents = new Student[]{  
 new Student("Dharma"),new Student("sanvi"),  
 new Student("Rupa"),new Student("Ajay")  
 };  
 // accessing the elements of the specified array  
 for(Student m:myStudents){  
 System.*out*.println(m);  
 }  
 }  
}

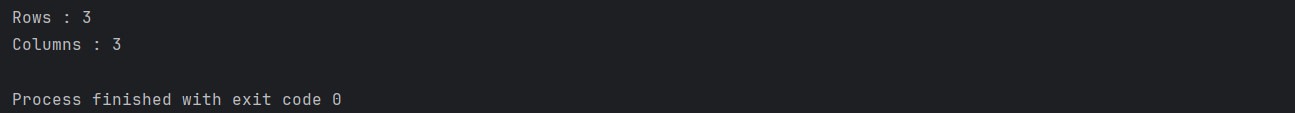


Task025:

public class Task025 {  
 public static void main(String[] args)  
 {  
 int[] arr = new int[4];  
 arr[0] = 10;  
 arr[1] = 20;  
 arr[2] = 30;  
 arr[3] = 40;  
  
 System.*out*.println(  
 "Trying to access element outside the size of array");  
 System.*out*.println(arr[5]);  
 }  
}



import java.io.\*;  
  
class Task025 {  
 public static void main(String[] args){  
  
 // Two Dimensional Array  
 // Declared and Initialized  
 int[][] arr = new int[3][3];  
  
  
 // Number of Rows  
 System.*out*.println("Rows : " + arr.length);  
  
 // Number of Columns  
 System.*out*.println("Columns : " + arr[0].length);  
 }  
}



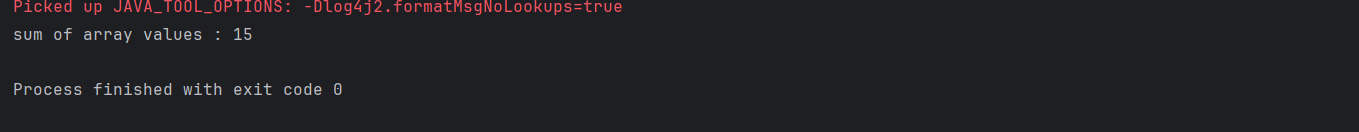
Task026:

public class Task026 {  
 // main function  
 public static void main(String args[])  
 {  
 // declaring and initializing 2D array  
 int arr[][] = { { 2, 7, 9 }, { 3, 6, 1 }, { 7, 4, 2 } };  
  
 // printing 2D array  
 for (int i = 0; i < 3; i++) { // rows  
 for (int j = 0; j < 3; j++) // columns  
 System.*out*.print(arr[i][j] + " ");  
  
 System.*out*.println();  
 }  
 }  
}



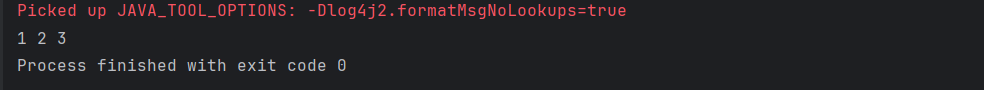
Task027:

public class Task027 {  
 // Driver method  
 public static void main(String args[])  
 {  
 int arr[] = { 3, 1, 2, 5, 4 };  
  
 // passing array to method m1  
 *sum*(arr);  
 }  
  
 public static void sum(int[] arr)  
 {  
 // getting sum of array values  
 int sum = 0;  
  
 for (int i = 0; i < arr.length; i++)  
 sum += arr[i];  
  
 System.*out*.println("sum of array values : " + sum);  
 }  
}



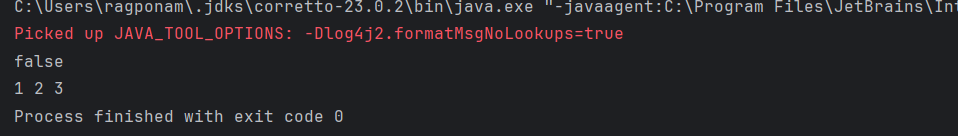
Task028:

class Test {  
 // Driver method  
 public static void main(String args[])  
 {  
 int arr[] = *m1*();  
  
 for (int i = 0; i < arr.length; i++)  
 System.*out*.print(arr[i] + " ");  
 }  
  
 public static int[] m1()  
 {  
 // returning array  
 return new int[] { 1, 2, 3 };  
 }  
}



Task029:

class Task029 {  
 public static void main(String args[])  
 {  
 int intArray[] = { 1, 2, 3 };  
  
 int cloneArray[] = intArray.clone();  
  
 // will print false as shallow copy is created  
 System.*out*.println(intArray == cloneArray);  
  
 for (int i = 0; i < cloneArray.length; i++) {  
 System.*out*.print(cloneArray[i] + " ");  
 }  
 }  
}



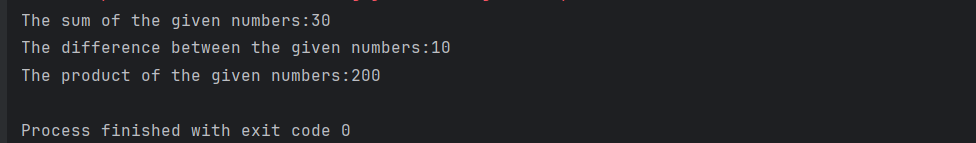
Task30:

class Task030 {  
 public static void main(String args[])  
 {  
 int intArray[][] = { { 1, 2, 3 }, { 4, 5 } };  
  
 int cloneArray[][] = intArray.clone();  
  
 // will print false  
 System.*out*.println(intArray == cloneArray);  
  
 // will print true as shallow copy is created  
 // i.e. sub-arrays are shared  
 System.*out*.println(intArray[0] == cloneArray[0]);  
 System.*out*.println(intArray[1] == cloneArray[1]);  
 }  
}



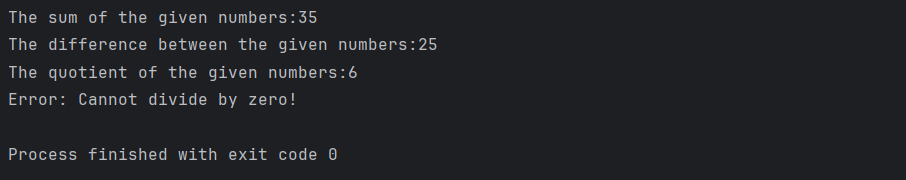
Task031:

class Calculation {  
 int z;  
  
 public void addition(int x, int y) {  
 z = x + y;  
 System.*out*.println("The sum of the given numbers:"+z);  
 }  
  
 public void Subtraction(int x, int y) {  
 z = x - y;  
 System.*out*.println("The difference between the given numbers:"+z);  
 }  
}  
  
public class Task031 extends Calculation {  
 public void multiplication(int x, int y) {  
 z = x \* y;  
 System.*out*.println("The product of the given numbers:"+z);  
 }  
  
 public static void main(String args[]) {  
 int a = 20, b = 10;  
 Task031 demo = new Task031();  
 demo.addition(a, b);  
 demo.Subtraction(a, b);  
 demo.multiplication(a, b);  
 }  
}  
  
class My\_Calculation2 extends Calculation {  
 public void multiplication(int x, int y) {  
 z = x \* y;  
 System.*out*.println("The product of the given numbers:"+z);  
 }  
  
 public static void main(String args[]) {  
 int a = 20, b = 10;  
 Task031 demo = new Task031();  
 demo.addition(a, b);  
 demo.Subtraction(a, b);  
 demo.multiplication(a, b);  
 }  
}



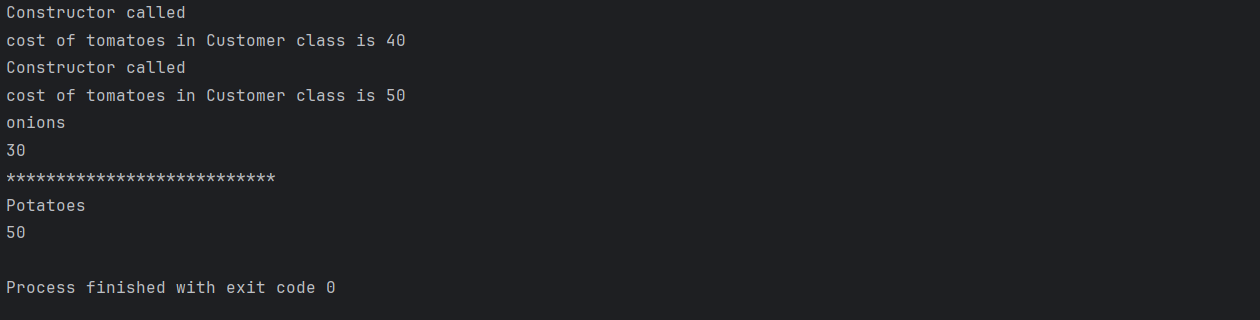
Task032:

class Calculation2 {  
 int z;  
  
 public void addition(int x, int y) {  
 z = x + y;  
 System.*out*.println("The sum of the given numbers:" + z);  
 }  
  
 public void Subtraction(int x, int y) {  
 z = x - y;  
 System.*out*.println("The difference between the given numbers:" + z);  
 }  
}  
  
public class Task032 extends Calculation {  
  
 public void division(int x, int y) {  
 if (y != 0) {  
 z = x / y;  
 System.*out*.println("The quotient of the given numbers:" + z);  
 } else {  
 System.*out*.println("Error: Cannot divide by zero!");  
 }  
 }  
  
 public static void main(String args[]) {  
 int a = 30, b = 5;  
  
 Task032 calculatorDemo = new Task032();  
  
 calculatorDemo.addition(a, b);  
 calculatorDemo.Subtraction(a, b);  
  
 calculatorDemo.division(a, b);  
  
 calculatorDemo.division(a, 0);  
 }  
}



Task033:

class Customer{  
 int cost = 40;  
 String items = "Tomatoes";  
 Customer(){  
 System.*out*.println("Constructor called");  
 }  
 void purchage\_list(){  
 System.*out*.println("cost of tomatoes in Customer class is "+ cost);  
 }  
}  
public class Task033 extends Customer {  
  
 void billing(){  
 String items = "onions";  
 int cost = 30;  
  
 super.items = "Potatoes";  
 super.cost = 50;  
 super.purchage\_list();  
  
 System.*out*.println(items);  
 System.*out*.println(cost);  
 System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  
 System.*out*.println(super.items);  
 System.*out*.println(super.cost);  
 // return 0;  
 }  
 public static void main(String[] args){  
 Customer cobj =new Customer();  
 cobj.purchage\_list();  
 Task033 tobj = new Task033();  
 tobj.billing();  
  
 }  
  
}



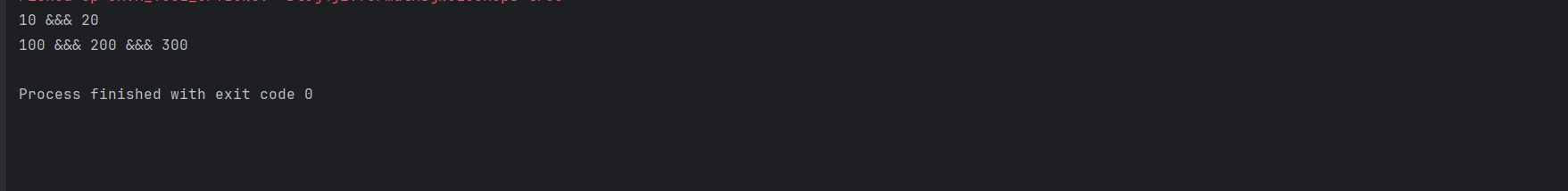
Task033\_1:

class Superclass{  
 int var;  
 Superclass(int var){  
 this.var = var;  
 }  
 public void getVar(){  
 System.*out*.println("var value in super class is "+ var);  
 }  
}  
  
public class Task033\_1 extends Superclass{  
 Task033\_1(int var) {  
 super(var);  
 }  
 public static void main(String[] args){  
 Superclass sobj = new Superclass(100);  
 sobj.getVar();  
 }  
}



Task034:

public class Task034{  
 public static void add(int x, int y){  
 System.*out*.println(x+ " &&& "+ y);  
 }  
 public static void add(int x, int y, int z ){  
 System.*out*.println(x + " &&& "+ y+ " &&& "+ z);  
 }  
 public static void main(String[] args){  
 Task034 tobj = new Task034();  
 *add*(10,20);  
 *add*(100,200,300);  
 }  
}



Task035:

public class Task035{  
 public static void add(int x, int y){  
 System.*out*.println(x + " %%%% "+ y);  
 }  
 public static void add(char x, char y){  
 System.*out*.println(x + " ###### "+ y);  
 }  
 public static void main(String[] args){  
 *add*(10,50);  
 *add*('R', 'R');  
 }  
}



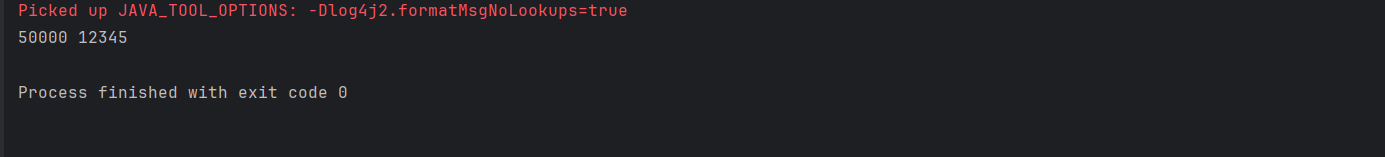
Task036:

public class Task036 {  
 public static void add(int x, float y){  
 System.*out*.println(x+" ||| "+y);  
 }  
 public static void add(float x, int y){  
 System.*out*.println(x+" }}{{ "+y);  
 }  
  
 public static void main(String[] args) {  
 *add*(10.50f, 60);  
 *add*(30,39.66f);  
 }  
}



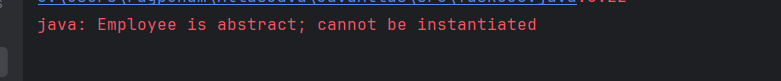
Task037:

class employee{  
 protected int salary;  
 public int emp\_id;  
 private int pwd;  
 }  
 class hr extends employee{  
 void details() {  
// super.pwd = 1234;  
 super.salary = 50000;  
 super.emp\_id = 12345;  
 System.*out*.println(salary+" "+emp\_id);  
 }  
  
 }  
 public class Task037 {  
 public static void main(String[] args) {  
 hr objhr = new hr();  
 objhr.details();  
  
  
 }  
}



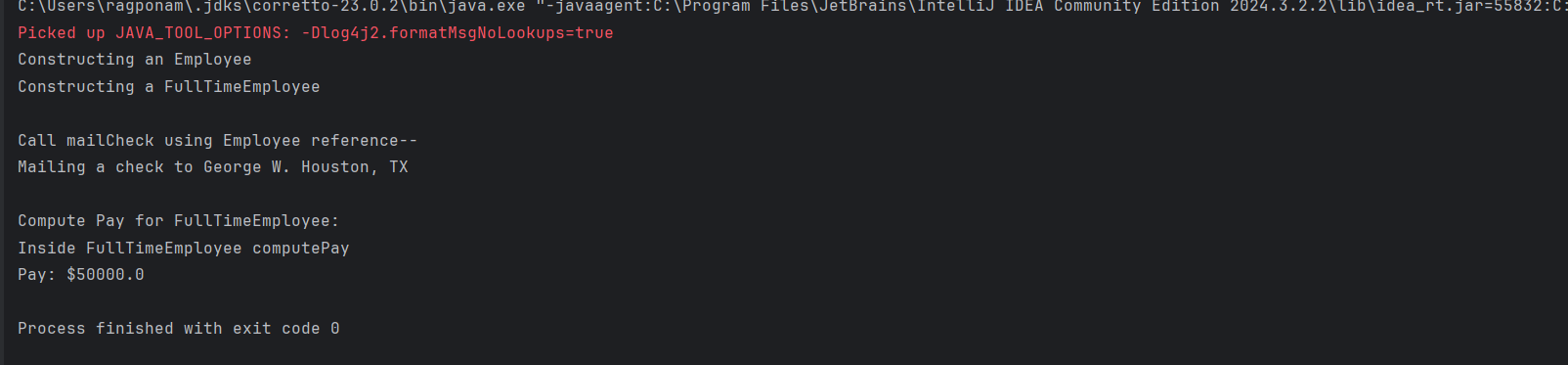
Task038:

public class Task038 {  
  
 public static void main(String [] args) {  
 /\* Following is not allowed and would raise error \*/  
 Employee e = new Employee("George W.", "Houston, TX", 43);  
 System.*out*.println("\n Call mailCheck using Employee reference--");  
 e.mailCheck();  
 }  
}  
  
abstract class Employee {  
 private String name;  
 private String address;  
 private int number;  
  
 public Employee(String name, String address, int number) {  
 System.*out*.println("Constructing an Employee");  
 this.name = name;  
 this.address = address;  
 this.number = number;  
 }  
  
 public double computePay() {  
 System.*out*.println("Inside Employee computePay");  
 return 0.0;  
 }  
  
 public void mailCheck() {  
 System.*out*.println("Mailing a check to " + this.name + " " + this.address);  
 }  
  
 public String toString() {  
 return name + " " + address + " " + number;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public String getAddress() {  
 return address;  
 }  
  
 public void setAddress(String newAddress) {  
 address = newAddress;  
 }  
  
 public int getNumber() {  
 return number;  
 }  
}



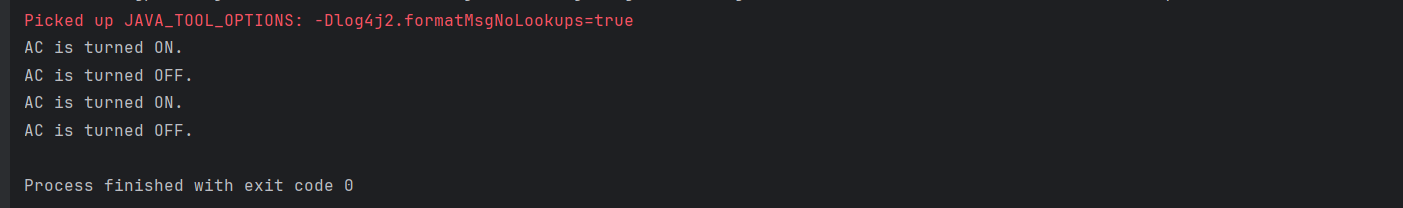
Task039:

public class Task039 {  
  
 public static void main(String [] args) {  
  
 FullTimeEmployee e = new FullTimeEmployee("George W.", "Houston, TX", 43, 50000.0);  
  
 System.*out*.println("\nCall mailCheck using Employee reference--");  
 e.mailCheck();  
  
 System.*out*.println("\nCompute Pay for FullTimeEmployee:");  
 System.*out*.println("Pay: $" + e.computePay());  
 }  
}  
  
abstract class Employee {  
 private String name;  
 private String address;  
 private int number;  
  
 public Employee(String name, String address, int number) {  
 System.*out*.println("Constructing an Employee");  
 this.name = name;  
 this.address = address;  
 this.number = number;  
 }  
  
 public double computePay() {  
 System.*out*.println("Inside Employee computePay (default calculation)");  
 return 0.0;  
 }  
  
 public void mailCheck() {  
 System.*out*.println("Mailing a check to " + this.name + " " + this.address);  
 }  
  
 @Override  
 public String toString() {  
 return "Name: " + name + ", Address: " + address + ", Number: " + number;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public String getAddress() {  
 return address;  
 }  
  
 public void setAddress(String newAddress) {  
 address = newAddress;  
 }  
  
 public int getNumber() {  
 return number;  
 }  
}  
  
class FullTimeEmployee extends Employee {  
 private double monthlySalary;  
  
 public FullTimeEmployee(String name, String address, int number, double monthlySalary) {  
 super(name, address, number);  
 this.monthlySalary = monthlySalary;  
 System.*out*.println("Constructing a FullTimeEmployee");  
 }  
  
 @Override  
 public double computePay() {  
 System.*out*.println("Inside FullTimeEmployee computePay");  
 return monthlySalary;  
 }  
  
 public double getMonthlySalary() {  
 return monthlySalary;  
 }  
  
 public void setMonthlySalary(double monthlySalary) {  
 this.monthlySalary = monthlySalary;  
 }  
}



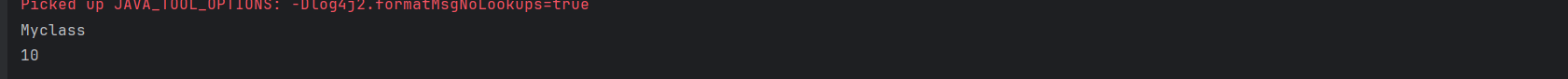
Task040:

abstract class Gadgets {  
 abstract void turnOn();  
 abstract void turnOff();  
}  
// Concrete class implementing the abstract methods  
class TVRemote extends Gadgets {  
 @Override  
 void turnOn() {  
 System.*out*.println("TV is turned ON.");  
 }  
  
 @Override  
 void turnOff() {  
 System.*out*.println("TV is turned OFF.");  
 }  
}  
class ACRemote extends Gadgets {  
 @Override  
 void turnOn() {  
 System.*out*.println("AC is turned ON.");  
 }  
  
 @Override  
 void turnOff() {  
 System.*out*.println("AC is turned OFF.");  
 }  
}  
  
// Main class to demonstrate abstraction  
public class Task040 {  
 public static void main(String[] args) {  
 Gadgets remote = new TVRemote();  
 remote = new ACRemote();  
 remote.turnOn();  
 remote.turnOff();  
  
// Gadgets remote = new FanRemote();  
// Gadgets remote = new CoolerRemote();  
 remote.turnOn();  
 remote.turnOff();  
 }  
}



Task041:

// Interface Declared  
//Driver Code Ends  
interface testInterface {  
 // public, static and final  
 final int *tax* = 10;  
 // public and abstract  
 void display();  
}  
  
  
// Class implementing interface  
class TestClass implements testInterface {  
 // Implementing the capabilities of  
 // Interface  
 public void display(){  
 System.*out*.println("Myclass");  
 }  
}  
class Myclass  
//Driver Code Starts  
{  
 public static void main(String[] args)  
 {  
 TestClass t = new TestClass();  
 t.display();  
 System.*out*.println(t.*tax*);  
 }  
  
  
}



Task042:

Final Vs. Constant: Final, when applied to a **variable**, it means the variable's value can be assigned only once. Once assigned, it cannot be changed (it becomes effectively "constant" in value). No specific keyword such as constant in java.