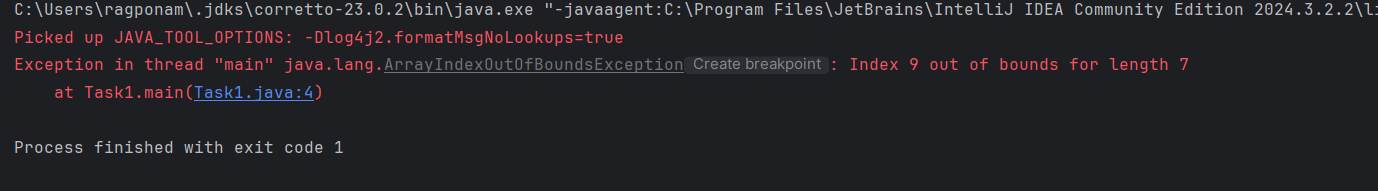
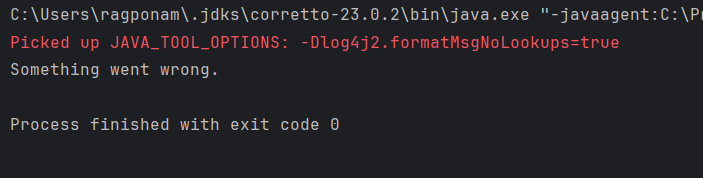
Task1:

public class Task1 {  
 public static void main(String[] args) {  
 int[] arr = {1, 2, 3, 4, 5, 6, 7};  
 System.*out*.println(arr[9]);  
 }  
}



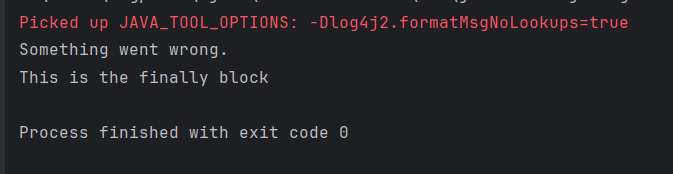
Task2:

public class Task2 {  
 public static void main(String[] args) {  
 try {  
 int[] myNums = {2, 4, 6};  
 System.*out*.println(myNums[5]);  
 }  
 catch (Exception e){  
 System.*out*.println("Something went wrong.");  
 }  
 }  
}



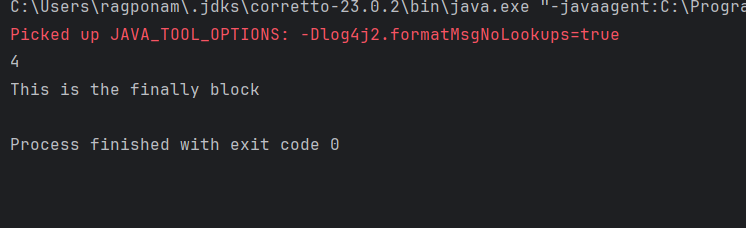
Task3:

public class Task3 {  
 public static void main(String[] args) {  
 try {  
 int[] myNumnbers = {2, 3, 4};  
 System.*out*.println(myNumnbers[5]);  
 }  
 catch (Exception e){  
 System.*out*.println("Something went wrong.");  
 }finally {  
 System.*out*.println("This is the finally block");  
 }  
 }  
}



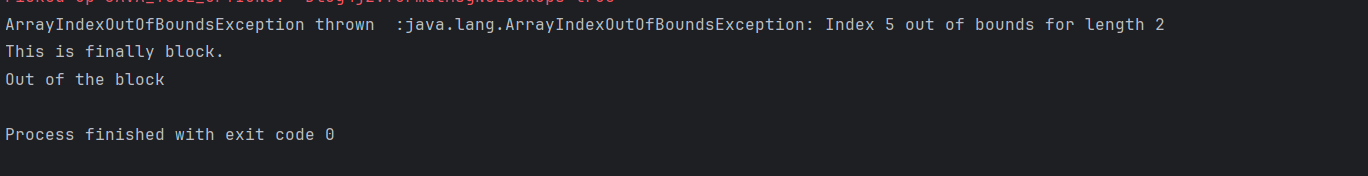
Task4:

public class Task3 {  
 public static void main(String[] args) {  
 try {  
 int[] myNumnbers = {2, 3, 4};  
 System.*out*.println(myNumnbers[2]);  
 }  
 catch (Exception e){  
 System.*out*.println("Something went wrong.");  
 }finally {  
 System.*out*.println("This is the finally block");  
 }  
 }  
}



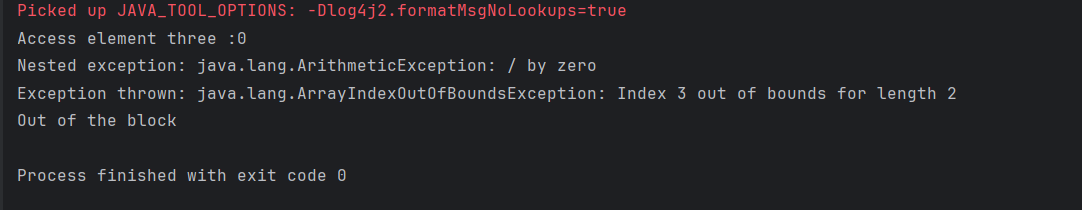
Task5:

public class Task5 {  
 public static void main(String[] args) {  
 try {  
 int a[] = new int[2];  
 int b = 10;  
 int c = 1/b;  
 int d = 0;  
 System.*out*.println("Access element three :" + a[5]);  
 System.*out*.println(b/d); // this line will not run as long as there is an exception in the previous line  
 }  
 catch (ArithmeticException e) {  
 System.*out*.println("ArithmeticException thrown :" + e);  
 }  
 catch (ArrayIndexOutOfBoundsException e) {  
 System.*out*.println("ArrayIndexOutOfBoundsException thrown :" + e);  
 }catch (Exception e) {  
 System.*out*.println("Exception thrown :" + e.getMessage());  
 }  
 finally {  
 System.*out*.println("This is finally block.");  
 }  
 System.*out*.println("Out of the block");  
  
 }  
}



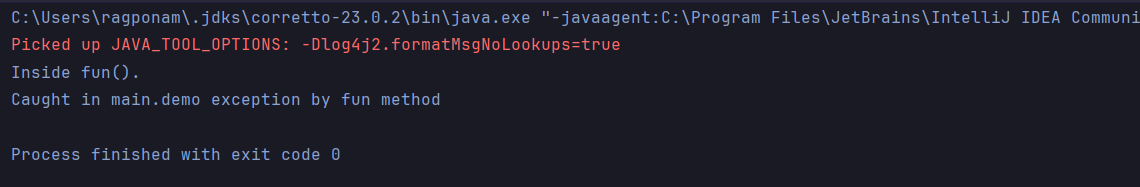
Task6:

public class Task6 {  
  
 public static void main(String args[]) {  
 try {  
 int a[] = new int[2];  
 System.*out*.println("Access element three :" + a[1]);  
 try {  
 int b = 0;  
 int c = 1/b;  
 }catch(Exception e) {  
 System.*out*.println("Nested exception: " + e);  
 }  
 System.*out*.println("Access element three :" + a[3]);  
 }  
 catch (ArrayIndexOutOfBoundsException e) {  
 System.*out*.println("Exception thrown: " + e);  
 }  
 System.*out*.println("Out of the block");  
 }  
}



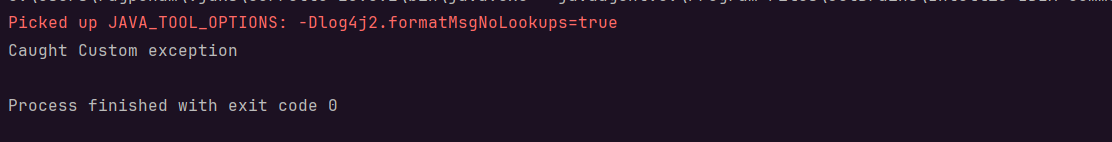
Task7:

class MyClass {  
  
 static void fun() throws IllegalAccessException  
 {  
 System.*out*.println("Inside fun(). ");  
 throw new IllegalAccessException("demo exception by fun method");  
 }  
  
 public static void main(String args[])  
 {  
 try {  
 *fun*();  
 //method2(); → arrayindex…  
 //Method3() —> file not found….  
 }  
 catch (IllegalAccessException e) {  
 System.*out*.println("Caught in main." + e.getMessage());  
 }  
 }  
}



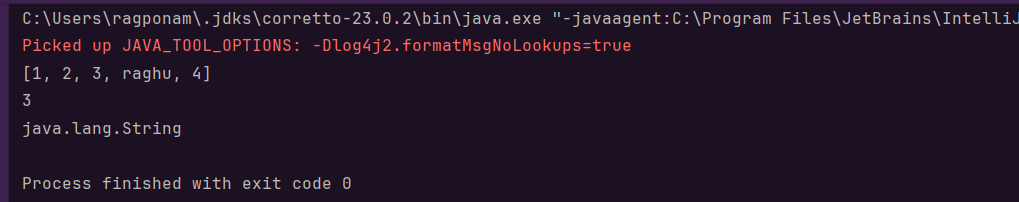
Task8:

class **Myexception** extends **Exception**{  
 public Myexception(**String** m){  
 super(m);  
 }  
}  
public class **Task8** {  
 public static void main(**String**[] args) {  
 try {  
 throw new Myexception("Custom exception");  
 } catch (**Myexception** e) {  
 **System**.*out*.println("Caught"+ " "+ e.getMessage());  
// System.out.print(e.getMessage());  
 }  
 }  
}



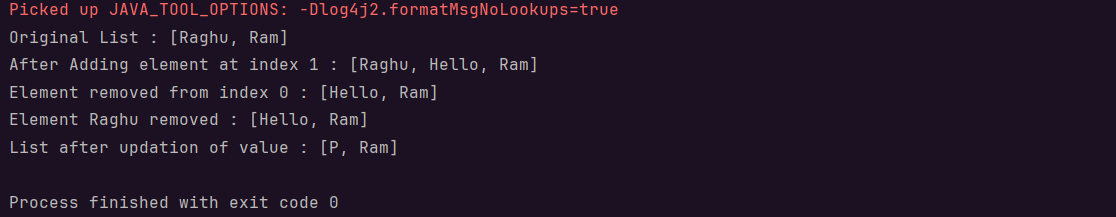
Task9:

import **java.util.ArrayList**;  
class **Main** {  
 public static void main(**String**[] args) {  
 **ArrayList**<**Object**> a = new ArrayList<>();  
 a.add(1);  
 a.add(2);  
 a.add(3);  
 a.add("raghu");  
 a.add("4");  
 **System**.*out*.println(a); //accessing the whole array list  
 **System**.*out*.println(a.get(2)); //accessing specific elements in an array list  
 **System**.*out*.println(a.get(4).getClass().getName());  
 }  
}



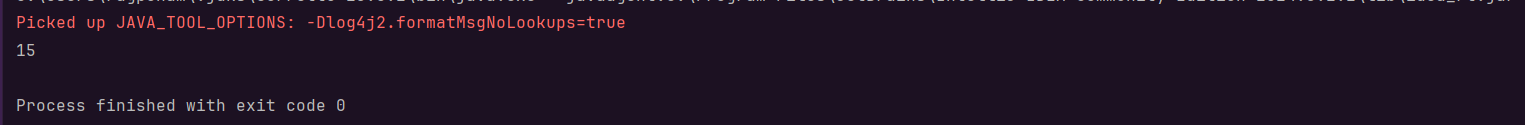
Task10:

import **java.util.**\*;  
class **Task10** {  
 public static void main(**String** args[]){  
 **ArrayList**<**String**> al = new ArrayList<>();  
 al.add("Raghu");  
 al.add("Ram");  
 **System**.*out*.println("Original List : "+al);  
  
 al.add(1, "Hello");  
 **System**.*out*.println("After Adding element at index 1 : "+ al);  
  
 al.remove(0);  
 **System**.*out*.println("Element removed from index 0 : "+ al);  
  
 al.remove("Raghu");  
 **System**.*out*.println("Element Raghu removed : "+ al);  
  
 al.set(0, "P");  
 **System**.*out*.println("List after updation of value : "+al);  
 }  
}



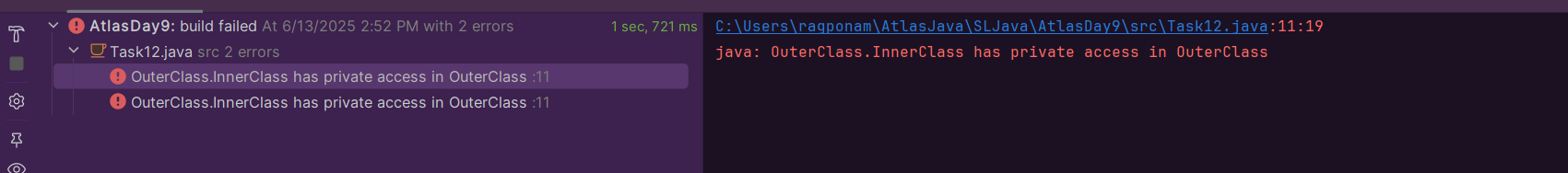
Task11:

class **OuterClass** {  
 int x = 10;  
 class **InnerClass** {  
 int y = 5;  
 }  
}  
  
public class **Task11** {  
 public static void main(**String**[] args) {  
 **OuterClass** myOuter = new OuterClass();  
 **OuterClass**.**InnerClass** myInner = myOuter.new InnerClass();  
 **System**.*out*.println(myInner.y + myOuter.x);  
 }  
}



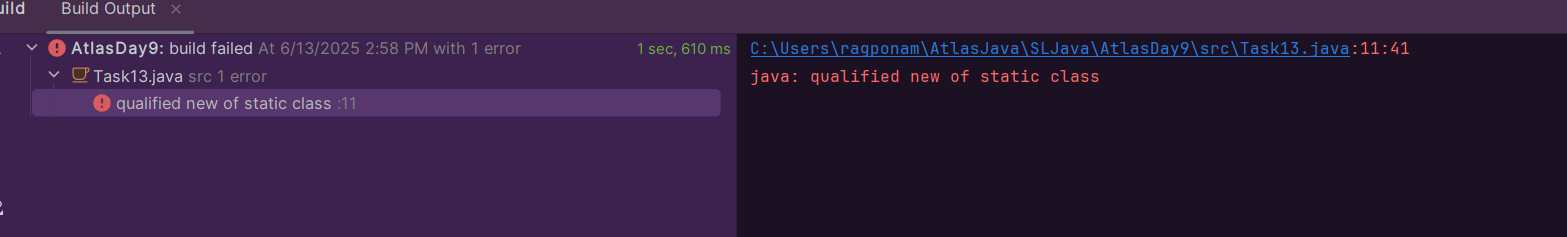
Task12:

class **OuterClass** {  
 int x = 10;  
 private class **InnerClass** {  
 int y = 5;  
 }  
}  
  
public class **Task12** {  
 public static void main(**String**[] args) {  
 **OuterClass** myOuter = new OuterClass();  
 **OuterClass**.**InnerClass** myInner = myOuter.new InnerClass();  
 **System**.*out*.println(myInner.y + myOuter.x);  
 }  
}

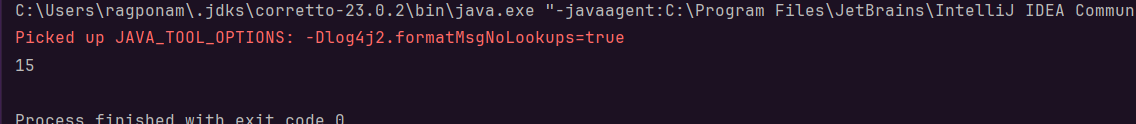


Task13:

class **OuterClass** {  
 int x = 10;  
 static class **InnerClass** {  
 int y = 5;  
 }  
}  
  
public class **Task13** {  
 public static void main(**String**[] args) {  
 **OuterClass** myOuter = new OuterClass();  
 **OuterClass**.**InnerClass** myInner = myOuter.new InnerClass();  
 **System**.*out*.println(myInner.y + myOuter.x);  
 }  
}

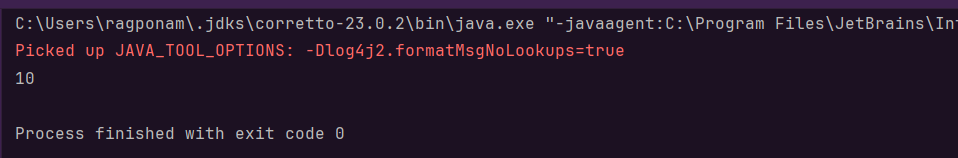


class **OuterClass** {  
 int x = 10;  
 static class **InnerClass** {  
 int y = 5;  
 }  
}  
  
public class **Task13** {  
 public static void main(**String**[] args) {  
 **OuterClass** myOuter = new OuterClass();  
 **OuterClass**.**InnerClass** myInner = new **OuterClass**.InnerClass();  
 **System**.*out*.println(myInner.y + myOuter.x);  
 }  
}



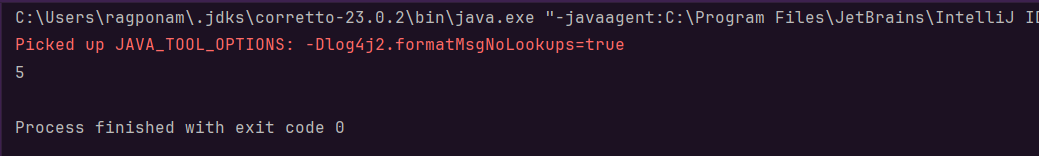
Task14:

class **OuterClass** {  
 int x = 10;  
 class **InnerClass** {  
 public int innerMethod(){  
 return x;  
 }  
 }  
}  
  
public class **Task14** {  
 public static void main(**String**[] args) {  
 **OuterClass** myOuter = new OuterClass();  
 **OuterClass**.**InnerClass** myInner = myOuter.new InnerClass();  
 **System**.*out*.println(myInner.innerMethod());  
 }  
}



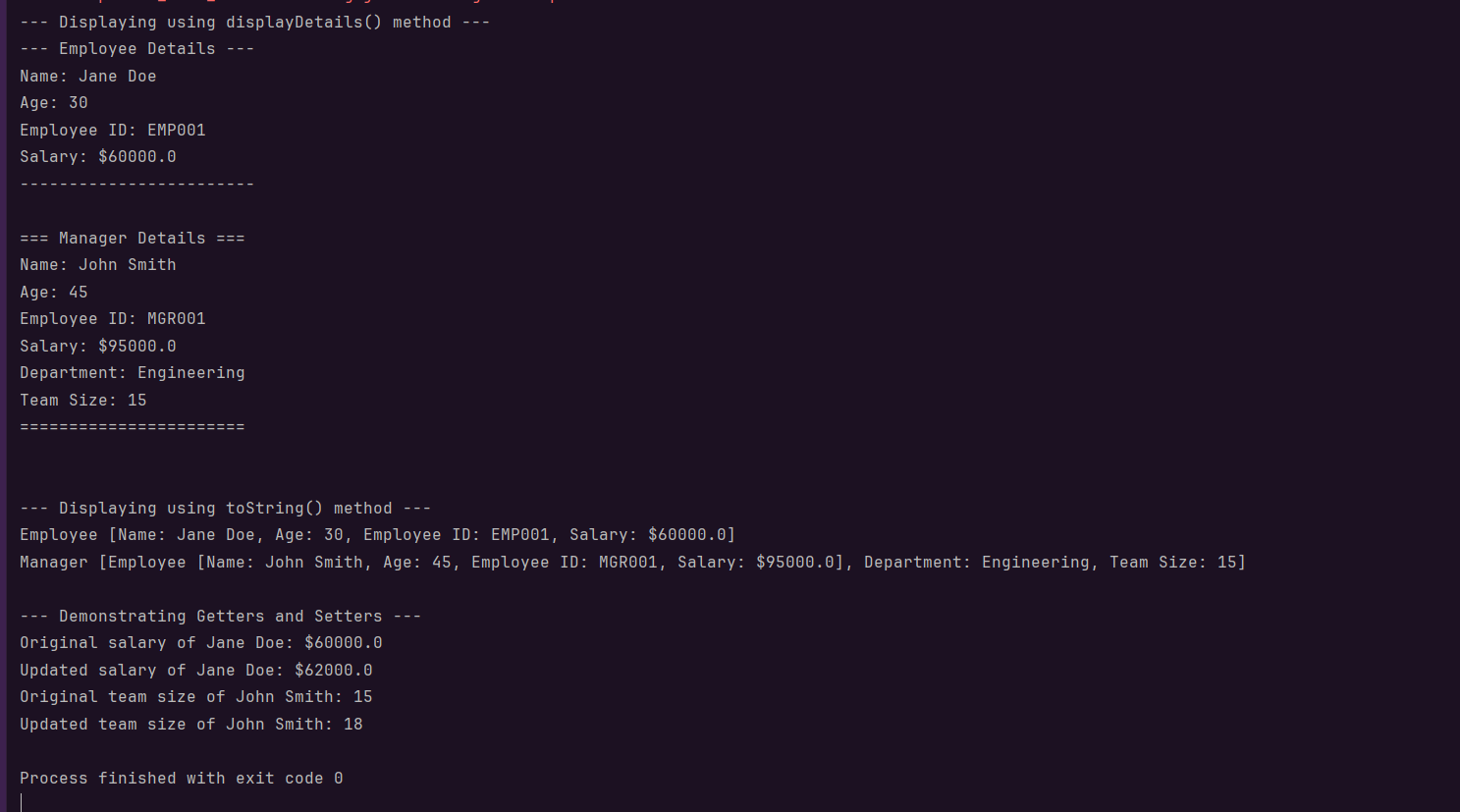
Task15:

class **OuterClass** {  
 int x = 10;  
 static class **InnerClass** {  
 static int *y* = 5;  
 }  
}  
public class **Task15** {  
 public static void main(**String**[] args) {  
 **OuterClass**.**InnerClass** myInner = new **OuterClass**.InnerClass();  
 **System**.*out*.println(myInner.*y*);  
  
 }  
}



Task16:

abstract class **Person** {  
 private **String** name;  
 private int age;  
  
 public Person(**String** name, int age) {  
 this.name = name;  
 this.age = age;  
 }  
  
 public **String** getName() {  
 return name;  
 }  
  
 public int getAge() {  
 return age;  
 }  
  
 public void setName(**String** name) {  
 this.name = name;  
 }  
  
 public void setAge(int age) {  
 this.age = age;  
 }  
  
 public abstract void displayDetails();  
  
 @Override  
 public **String** toString() {  
 return "Name: " + name + ", Age: " + age;  
 }  
}  
  
class **Employee** extends **Person** {  
 private **String** employeeId;  
 private double salary;  
  
 public Employee(**String** name, int age, **String** employeeId, double salary) {  
 super(name, age);  
 this.employeeId = employeeId;  
 this.salary = salary;  
 }  
  
 public **String** getEmployeeId() {  
 return employeeId;  
 }  
  
 public double getSalary() {  
 return salary;  
 }  
  
 public void setEmployeeId(**String** employeeId) {  
 this.employeeId = employeeId;  
 }  
  
 public void setSalary(double salary) {  
 this.salary = salary;  
 }  
  
 @Override  
 public void displayDetails() {  
 **System**.*out*.println("--- Employee Details ---");  
 **System**.*out*.println("Name: " + getName());  
 **System**.*out*.println("Age: " + getAge());  
 **System**.*out*.println("Employee ID: " + employeeId);  
 **System**.*out*.println("Salary: $" + salary);  
 **System**.*out*.println("------------------------");  
 }  
  
 @Override  
 public **String** toString() {  
 return "Employee [" + super.toString() + ", Employee ID: " + employeeId + ", Salary: $" + salary + "]";  
 }  
}  
  
class **Manager** extends **Employee** {  
 private **String** department;  
 private int teamSize;  
  
 public Manager(**String** name, int age, **String** employeeId, double salary, **String** department, int teamSize) {  
 super(name, age, employeeId, salary);  
 this.department = department;  
 this.teamSize = teamSize;  
 }  
  
 public **String** getDepartment() {  
 return department;  
 }  
  
 public int getTeamSize() {  
 return teamSize;  
 }  
  
 public void setDepartment(**String** department) {  
 this.department = department;  
 }  
  
 public void setTeamSize(int teamSize) {  
 this.teamSize = teamSize;  
 }  
  
 @Override  
 public void displayDetails() {  
 **System**.*out*.println("=== Manager Details ===");  
 **System**.*out*.println("Name: " + getName());  
 **System**.*out*.println("Age: " + getAge());  
 **System**.*out*.println("Employee ID: " + getEmployeeId());  
 **System**.*out*.println("Salary: $" + getSalary());  
 **System**.*out*.println("Department: " + department);  
 **System**.*out*.println("Team Size: " + teamSize);  
 **System**.*out*.println("=======================");  
 }  
  
 @Override  
 public **String** toString() {  
 return "Manager [" + super.toString() + ", Department: " + department + ", Team Size: " + teamSize + "]";  
 }  
}  
  
public class **Task016** {  
 public static void main(**String**[] args) {  
 **Employee** emp1 = new Employee("Jane Doe", 30, "EMP001", 60000.00);  
  
 **Manager** mgr1 = new Manager("John Smith", 45, "MGR001", 95000.00, "Engineering", 15);  
  
 **System**.*out*.println("--- Displaying using displayDetails() method ---");  
 emp1.displayDetails();  
 **System**.*out*.println();  
 mgr1.displayDetails();  
 **System**.*out*.println("**\n**");  
  
 **System**.*out*.println("--- Displaying using toString() method ---");  
 **System**.*out*.println(emp1);  
 **System**.*out*.println(mgr1);  
  
 **System**.*out*.println("**\n**--- Demonstrating Getters and Setters ---");  
 **System**.*out*.println("Original salary of Jane Doe: $" + emp1.getSalary());  
 emp1.setSalary(62000.00);  
 **System**.*out*.println("Updated salary of Jane Doe: $" + emp1.getSalary());  
  
 **System**.*out*.println("Original team size of John Smith: " + mgr1.getTeamSize());  
 mgr1.setTeamSize(18);  
 **System**.*out*.println("Updated team size of John Smith: " + mgr1.getTeamSize());  
 }  
}



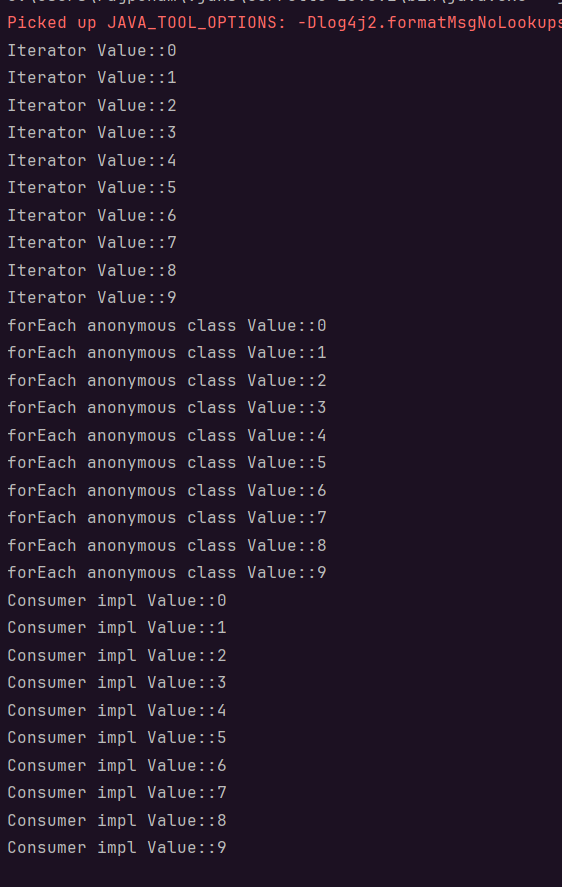
Task017:

Java 8 Features:

* Lambda expressions.
* Interfaces can have body (default and static)
* For each loops
* Collection API
* Stream API
* Java io improvements
* Functional Interfaces
* Method references
* Time API

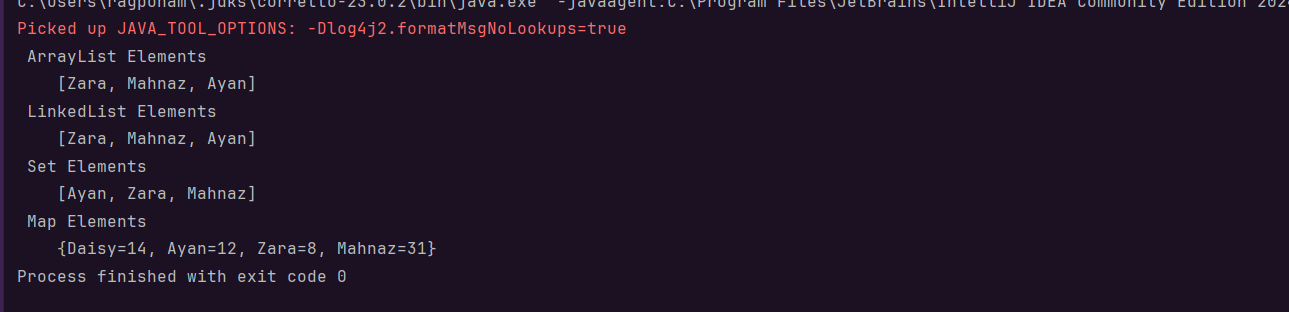
Task018:

import **java.util.ArrayList**;  
import **java.util.Iterator**;  
import **java.util.List**;  
import **java.util.function.Consumer**;  
import **java.lang.Integer**;  
  
public class **Task18** {  
  
 public static void main(**String**[] args) {  
  
 **List**<**Integer**> myList = new ArrayList<**Integer**>();  
 for(int i=0; i<10; i++) myList.add(i);  
  
 **Iterator**<**Integer**> it = myList.iterator();  
 while(it.hasNext()){  
 **Integer** i = it.next();  
 **System**.*out*.println("Iterator Value::"+i);  
 }  
  
 myList.forEach(new **Consumer**<**Integer**>() {  
  
 public void accept(**Integer** t) {  
 **System**.*out*.println("forEach anonymous class Value::"+t);  
 }  
  
 });  
  
 //traversing with Consumer interface implementation  
 **MyConsumer** action = new MyConsumer();  
 myList.forEach(action);  
  
 }  
  
}  
  
//Consumer implementation that can be reused  
class **MyConsumer** implements **Consumer**<**Integer**>{  
  
 public void accept(**Integer** t) {  
 **System**.*out*.println("Consumer impl Value::"+t);  
 }  
}



Task020:

import **java.util.**\*;  
public class **Task20** {  
  
 public static void main(**String**[] args) {  
 // ArrayList  
 **List** a1 = new ArrayList();  
 a1.add("Zara");  
 a1.add("Mahnaz");  
 a1.add("Ayan");  
 **System**.*out*.println(" ArrayList Elements");  
 **System**.*out*.print("**\t**" + a1);  
  
 // LinkedList  
 **List** l1 = new LinkedList();  
 l1.add("Zara");  
 l1.add("Mahnaz");  
 l1.add("Ayan");  
 **System**.*out*.println();  
 **System**.*out*.println(" LinkedList Elements");  
 **System**.*out*.print("**\t**" + l1);  
  
 // HashSet  
 **Set** s1 = new HashSet();  
 s1.add("Zara");  
 s1.add("Mahnaz");  
 s1.add("Ayan");  
 **System**.*out*.println();  
 **System**.*out*.println(" Set Elements");  
 **System**.*out*.print("**\t**" + s1);  
  
 // HashMap  
 **Map** m1 = new HashMap();  
 m1.put("Zara", "8");  
 m1.put("Mahnaz", "31");  
 m1.put("Ayan", "12");  
 m1.put("Daisy", "14");  
 **System**.*out*.println();  
 **System**.*out*.println(" Map Elements");  
 **System**.*out*.print("**\t**" + m1);  
 }  
}



Task17:

* Lambda Expressions
* Functional Interfaces
* Stream API.
* Default and Static method interfaces
* forEach()
* Collections
* Time API
* Method references
* Java io improvements

Task18:

The give program has methods to loop/iterate a given thread for 10 times using

* for loop – pre java8
* While loop – alternative-for pre java8
* forEach – introduced in java8, also known as advanced for loop

Task20: