Name : Gautam K K Date: 20/11/24

JAVA

Latest Features of Java 23

- introduces the generational mode for the Z Garbage Collector (ZGC)
- Enhanced Primitive Type Patterns
- introduces flexible constructor bodies
- Simplified Module Imports
- enhanced JDK 23 I/O API is designed for file and network I/O operations.
- Structured concurrency in JDK 23 aims to simplify error handling and improve reliability in concurrent programming.

Benefits of Inheritance

- Code Reusability
- Ease of Maintenace
- Improved Code Organization
- Dynamic Extension of Functionality
- Reduce redundancy

Constructors in Java

It is the special method in java which is used to initialize the objects

At the time of calling , the memory for the object is created in the memory

When the object is created with the new keyword any one constructor will be called

Constructor should not return any return type

TYPES OF CONSTRUCTORS

- Default constructor
- Parameterized constructor

Parameterized Constructor

The parameterized constructor id used to provide different values to the specific object created

EX: Student s1 = new Student(“gautam”,1913119);

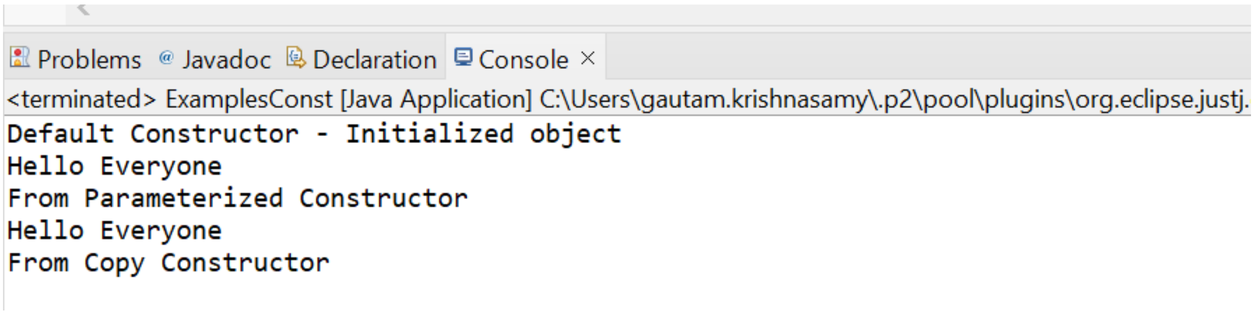
Copy constructor

Java supports a copy constructor but it is not predefined we want to manually declare

Code

```java
class Demo{

String msg;
Demo(){
System.out.println("Default Constructor - Initialized object");
}

Demo(String msg){
this.msg= msg;
System.out.println(msg);
System.out.println("From Parameterized Constructor");
}

Demo(Demo d){
this.msg = d.msg;
System.out.println(this.msg);
System.out.println("From Copy Constructor");
}
}

public class ExamplesConst {

public static void main(String[] args) {

//            creating object with different constructors

Demo d1 = new Demo();
Demo d2 = new Demo("Hello Everyone");
Demo d3 = new Demo(d2);
}
}
```

Output



```
<terminated> ExamplesConst [Java Application] C:\Users\gautam.krishnasamy\.p2\pool\plugins\org.eclipse.justj.
Default Constructor - Initialized object
Hello Everyone
From Parameterized Constructor
Hello Everyone
From Copy Constructor
```

Access Modifiers

There are four access modifiers

- Default
- Public
- Private
- Protected

Default

When class, methods, data members are not specified with any access modifiers then it will be default

Accessible only within the same package, within the class, outside the class

Public

- Public specifiers had the widest scope among the other specifiers
- Public class, methods, and data members can be accessible from anywhere in the program
- Accessible – within the class, outside the class, outside the package

Private

- Private methods, data members, inner class can be declared with the keyword private
- Accessible – only within the class, not outside the class, not outside the package

Protected

- Accessible – within the same class, outside the class of same package, or the class which extends this class within the same or other package
- Private, protected can be used with methods, data members & nested classes.

Public static void main()

Main() method is the entry point for the JVM to execute the program, if main() is missing it not execute

The syntax of main method should be in way so that, JVM will understand it should be public static void main

Public – it should be in public so that it is globally available, so that JVM can invoke the main method from outside the class

Static – keyword making it class related method, so that JVM can invoke the main method without creating the objects

This save unnecessary wastage of memory

Void

This keyword is used to specify that method doesn't return any return type

Since main is the end method in the execution if it returns also, it is not going anywhere

main should be in void return type.

Main

It is the identifier or keyword that JVM is to start executing the program

It should be in main predefined name, other names can't be work

Inner Class

Inner class is the class that declare inside the class or interface

Inner classes is used to logically group classes and interfaces in one place

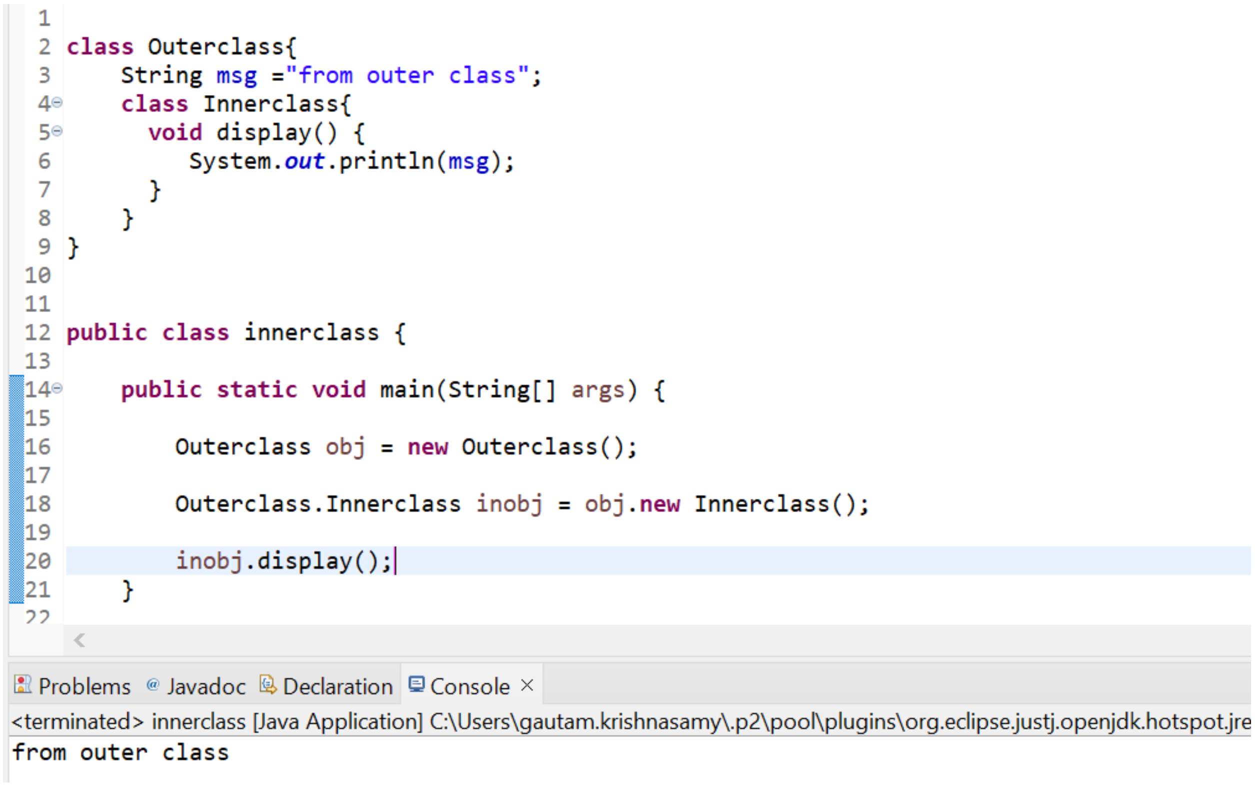
Nested classes can access all the data members and method of outer class including private

Non-Static Inner class

For accessing the inner class we need outer class with dot operator

And for creating inner class object, first need outer class object

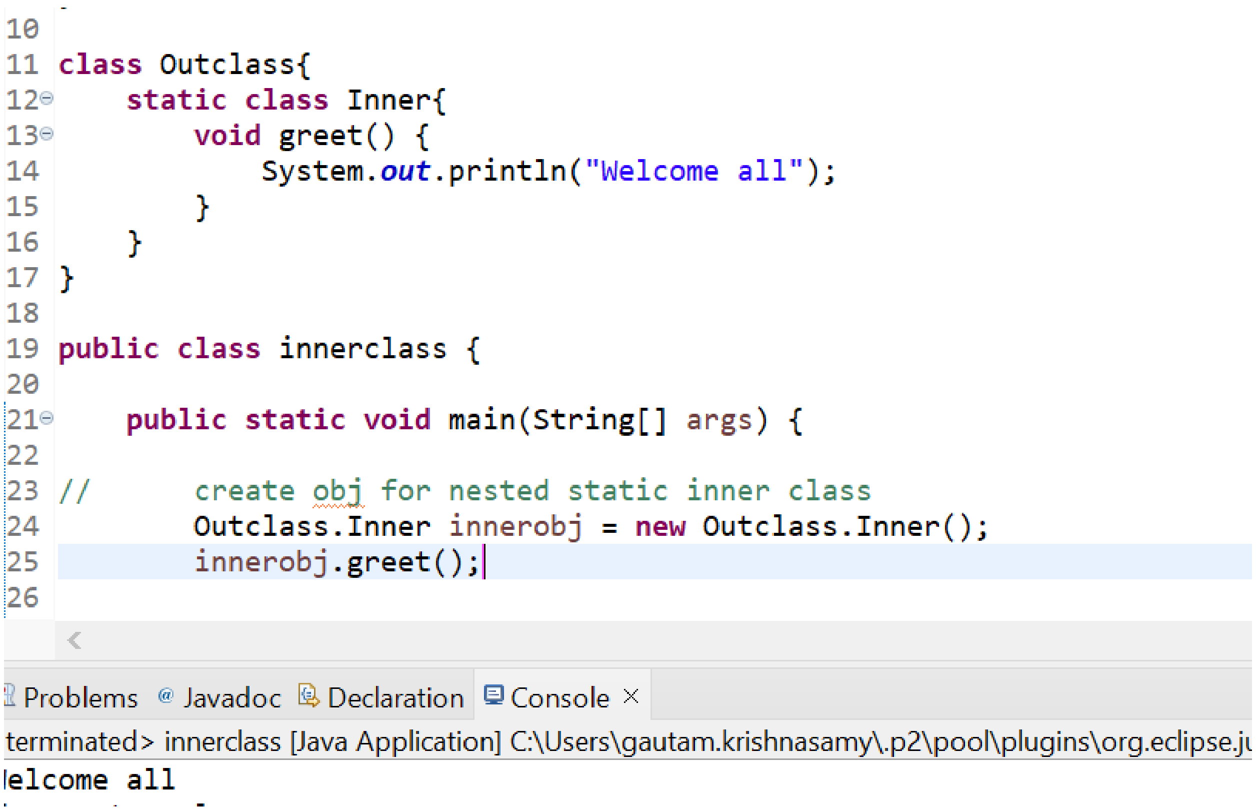
Ex:



```java

class Outerclass{
    String msg ="from outer class";
    class Innerclass{
      void display() {
         System.out.println(msg);
      }
    }
}


public class innerclass {

    public static void main(String[] args) {

        Outerclass obj = new Outerclass();

        Outerclass.Innerclass inobj = obj.new Innerclass();

        inobj.display();
    }
```

```
<terminated> innerclass [Java Application] C:\Users\gautam.krishnasamy\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre
from outer class
```

Static Inner class



```java

class Outclass{
    static class Inner{
        void greet() {
            System.out.println("Welcome all");
        }
    }
}

public class innerclass {

    public static void main(String[] args) {

//      create obj for nested static inner class
        Outclass.Inner innerobj = new Outclass.Inner();
        innerobj.greet();

```

Problems Javadoc Declaration Console

```
terminated> innerclass [Java Application] C:\Users\gautam.krishnasamy\.p2\pool\plugins\org.eclipse.ju
Jelcome all
```

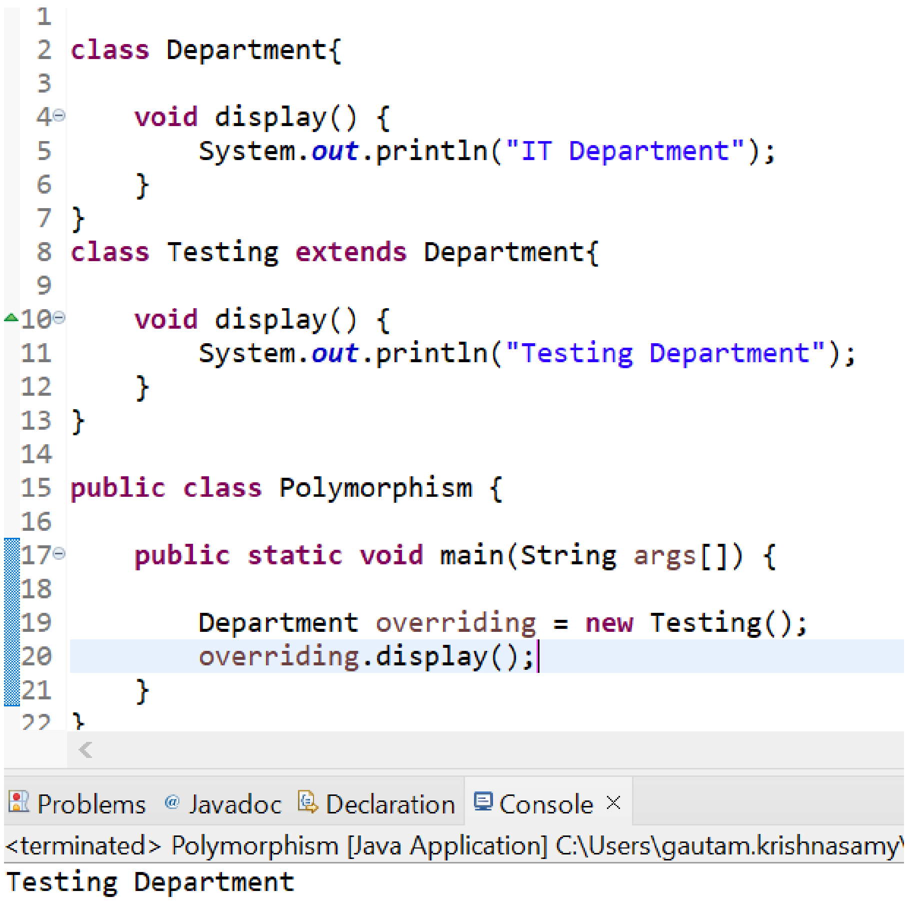
Variables in Abstract class & Interfaces

Abtract class –variables be private, protected, public, final, static

Interfaces – variables be public, static, final – cannot be change once initialized

Polymorphism

Method overriding



```

class Department{

    void display() {
        System.out.println("IT Department");
    }
}
class Testing extends Department{

    void display() {
        System.out.println("Testing Department");
    }
}

public class Polymorphism {

    public static void main(String args[]) {

        Department overriding = new Testing();
        overriding.display();
    }
}
```

```
<terminated> Polymorphism [Java Application] C:\Users\gautam.krishnasamy\
Testing Department
```