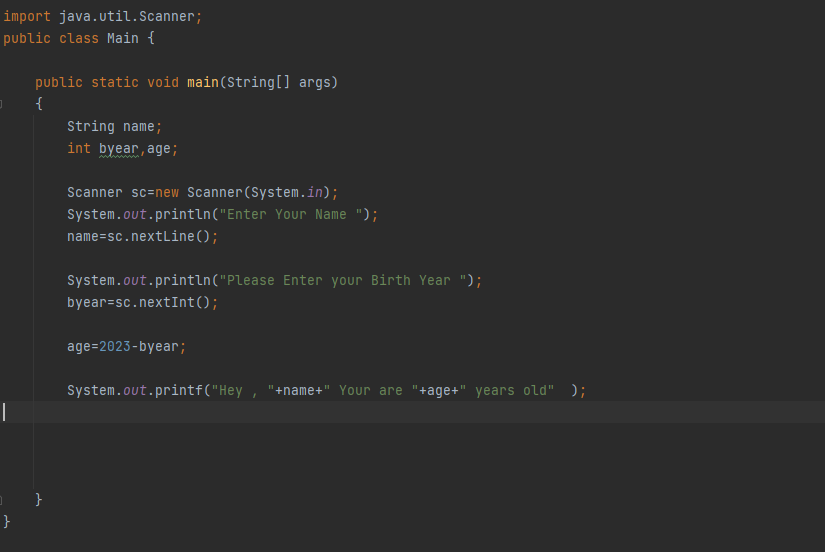
**2023.04.25**



**Working with Classes**

<syntax> - How to create a New Class?

* Class <class name >

{  
 // input data such as integers and Strings, etc…

//Methods

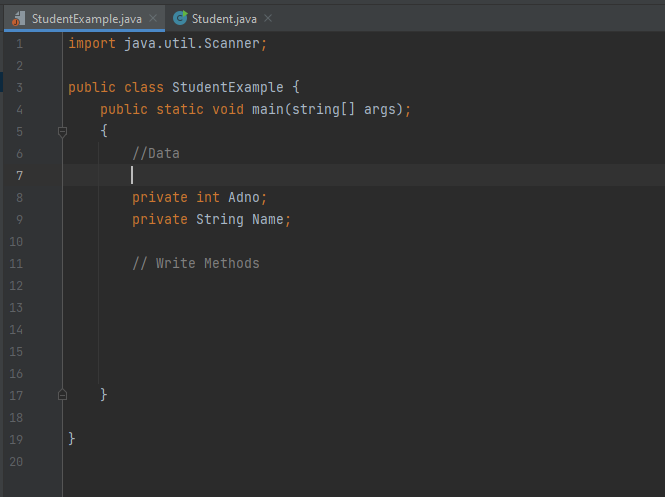
}

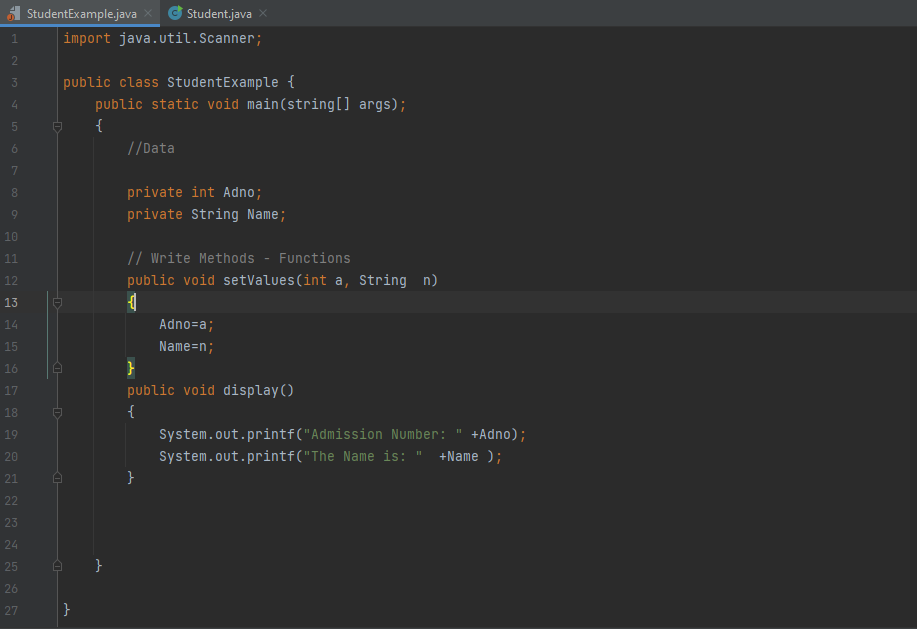
Create a Java Class Student with a following data and methods.

Data : Admission no, Name,

Methods: A method which accept two parameters and set the values

A Method to display data.





How to Create an object and Call the Methods ?

// which will be done inside the main method.,

<syntax>

Class <class Name>

{

Main method ()

{

<class name> <objectname> new <classname > ( ) ;

<objectname>.methodname();

}

}

Question

Write a java program to create the following class.

Width

Height

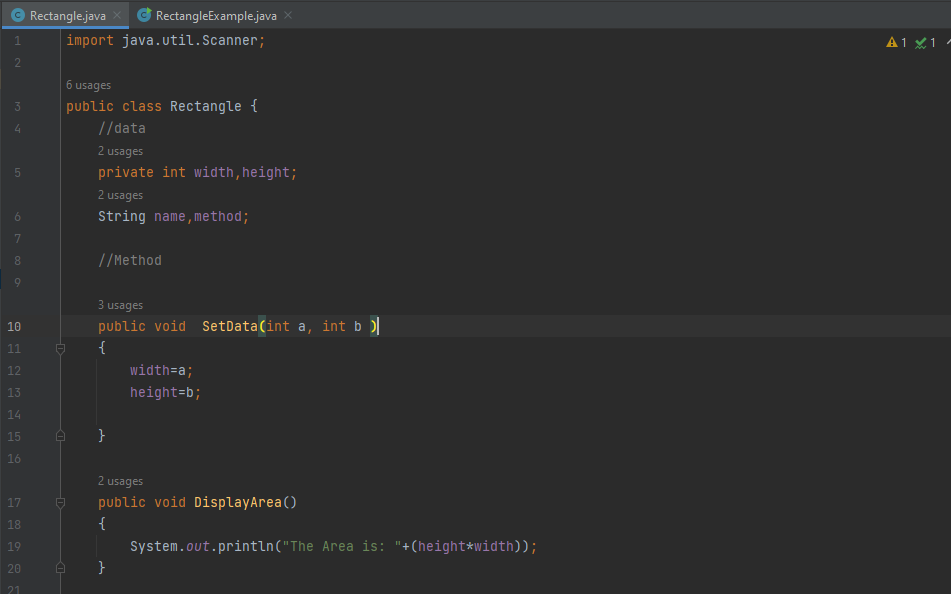
Rectangle

Set Data

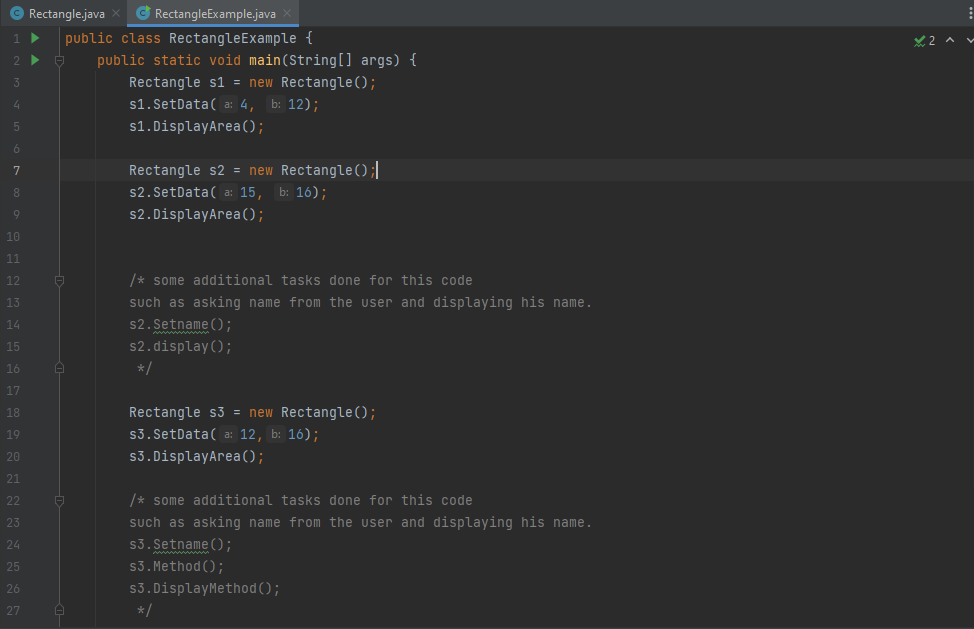
Display text

After Creating the class. Create two objects from the rectangle class and display the area of each of the rectangle.

Class – Rectangle // shows data and methods.



Object code – Main // shows function recalling.



// private – private data cannot be access these from the outside of the class with giving an error as a result.

* Inside a class usually the data is declared as private. it means you cannot access private data or private methods outside the class. (It can be accessed only inside the class where it has been declared)
* The public data or public methods are accessible inside and outside the class.

**2023/05/08**

Previous day was a holiday.

Example – to revise the last session.

* Create a class student with the given data and methods.

Data: Admission No, Name, Batch,

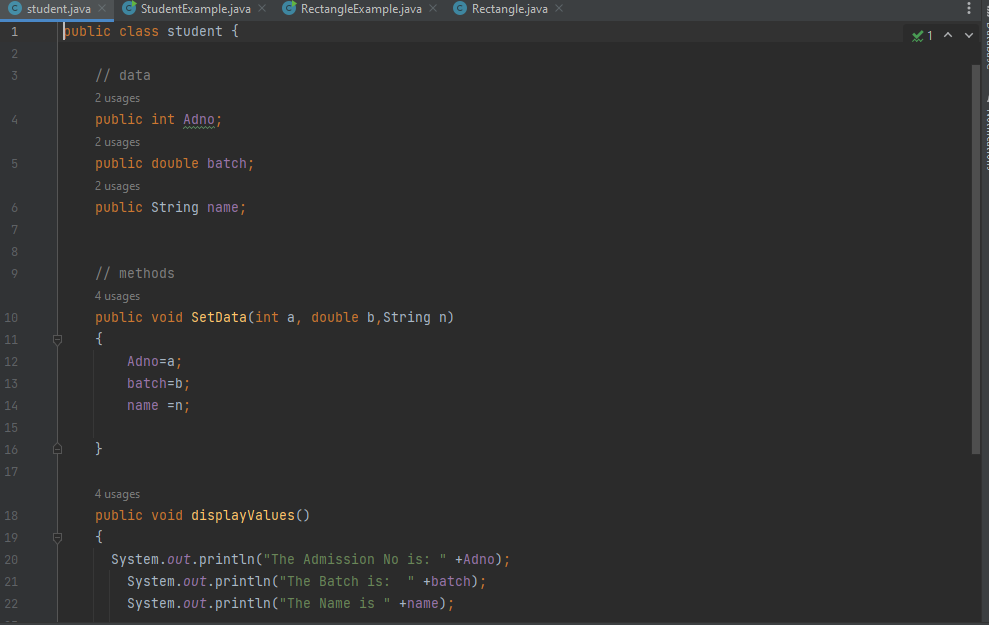
Methods: A method to set values for the above variables using parameters

A method to display the values of all the above variables.

The Answer for the Above question is as Follows:

The Class – Which is student.

/\* In the Code I have modified it with displaying the age after giving parameter birth year \*/



The Object Code – Student Example

A screenshot of a computer program

Description automatically generated with medium confidence

**Summary**

* Create the Class (Include the data and Methos)
* In the main class, Create the object.
* Call the function on the main object file (method).

Example 2:

**Constructer**

A Constructor is used to set values for the data or member variables of a class (it is a method).

In a class a constructor has the following characteristics

1. A constructor name is same name as class name.
2. It doesn’t have a return type.
3. There are parametrized and no parameter constructors.

Example 1

The Following class includes no parameter constructer.

Public class A

{

Private int a,b;

Public A()

{

A=-5;

B=10;

}

Public void display ()

{

System.out.println(“a” +a);

System.out.println(“b” +b);

}

The above constructor can be called inside the main class as follows.

Public class Abj1

{

Public static void main(String[]args)

{

A a1= new A();

//once you create an object the no parameter

//constructer will be automatically involved

A1. display(); // a is in simple

}

}

\*\*\*\*\* To the above class we can include parameterized constructer as follows:

Use the picture in the mobile

Parametrized – highlight part

The above parametrized constructer is called inside the main method as follows:

A a1=new A(200,300);

A1. display(); // a is in simple

Question

1. Create a class bank account.
2. Include an account number as a data Account Holders Name, Type and balance as variables to the above bank account class.
3. Include a method to set values using parameters.
4. Include the no parameter constructer and parametrized constructer to the above class.
5. Include a display method to the above class.
6. Inside the main class create an object and call all the above methods.

Class BankAccount,

{

//data

Private int Accountnumber, balance;

Private String Hname, type;

//method

Void Setvalues(int a, int b, String c, String d)

{

Accountnumber = a;

balance=b;

Hname = c;

type = d;

}

// no parameter Constructer

Public BankAccount()

{

Accountnumber = 0024578921826;

balance= 155555555555555;

Hname = Sadeesha;

type = Savings;

}

// Paramerized constructer

Public BankAccount(int x, 155555555555555,”Sadeesha”,”Savings”)

{

}

Void display()

{

System.out.println(“Account Number “ +Accountnumber);

System.out.println(“The balance is “ +balance);

System.out.println(“Accont Holders Name is: “ +Hname);

System.out.println(“The Account Type “ +type);

}

}

**2023.05.16**

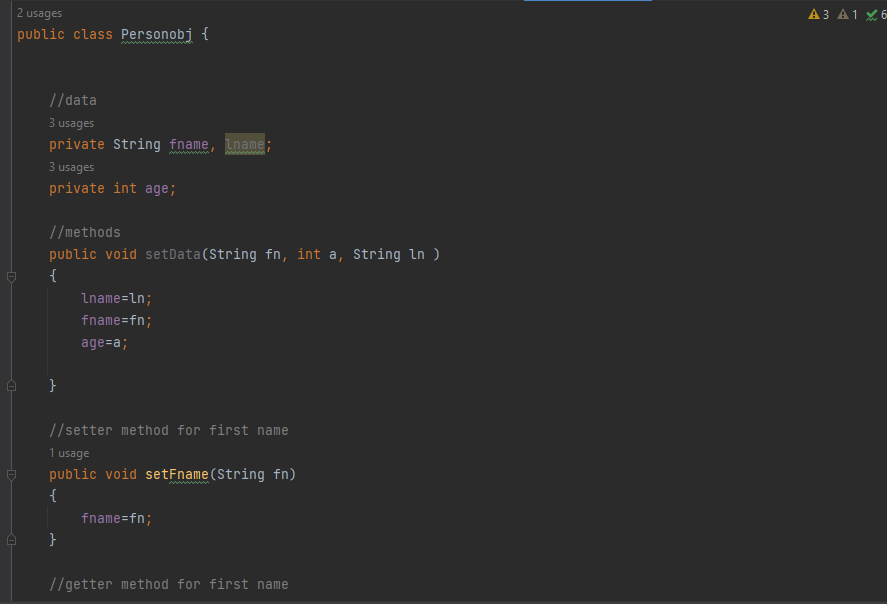
**ENCAPSULATION**

* Encapsulation is one of the fundamental OOP concepts.
* The meaning of encapsulation refers to the data hiding. Data hiding means in a class you declare the data as private and methods public. Therefore, private members are not accessible outside the class. This is called data Hiding.
* To achieve this, you must
* Declare class variables as private.
* Provide public get and set methods to access and update the value of a private variable.

**Getter & Setter Methods**

* The above methods are used to set and return values for the each of the member variables separately and return the values.

**The Class Code – Personobj**



A screenshot of a computer

Description automatically generated

**The Object Code – Person**

A picture containing text, screenshot, display, software

Description automatically generated

**INHERITANCE**

* In inheritance we create a relationship between the classes. In the following example the class A is parent of super class, and the class B is child of sub class.
* There is a single inheritance between the classes.
* As a result, class B can access class A’s public properties.
* We can only access private data on the parent class. Wont able to access private data on the parent class.

**SYNTAX**

Class <Parent Class>

{

}

Class <Child Class> extends <Parent Class>

{

}

**Example**

Create a following class structure using a java programme. Create an object from the child class and call parent class and child class methods.

**Parent Class**

Student

Admno,Name

setData (int,string)

displayData()

]

**Child Class**

Marks

Mark1, Mark2

setMarks(int,int)

displayMarks()

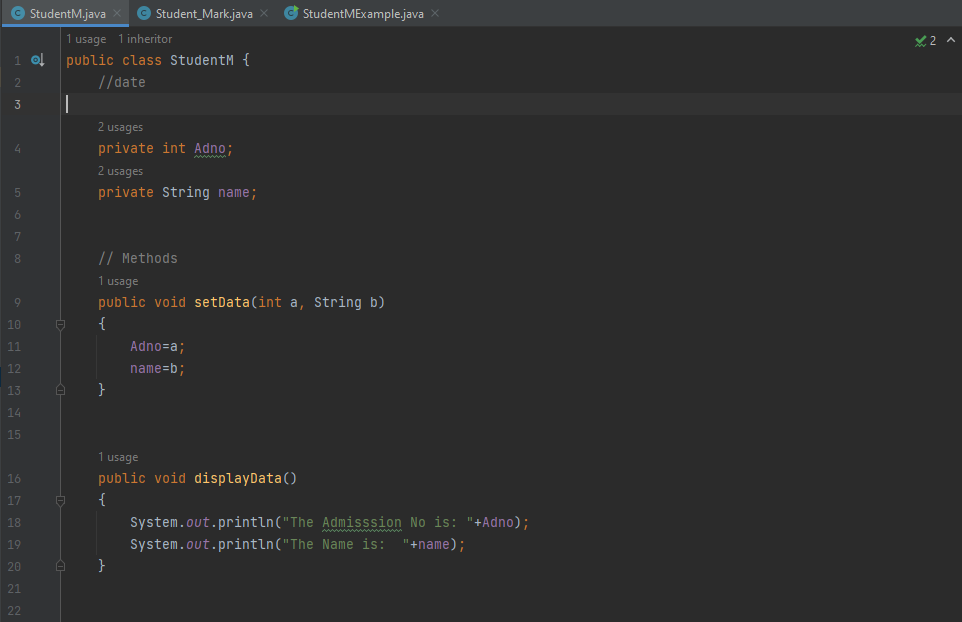
//The Example code of the above example can be found in the below screenshots

// it shows two classes one is StudentM which shows data for Admission no, name with methods to show data and display

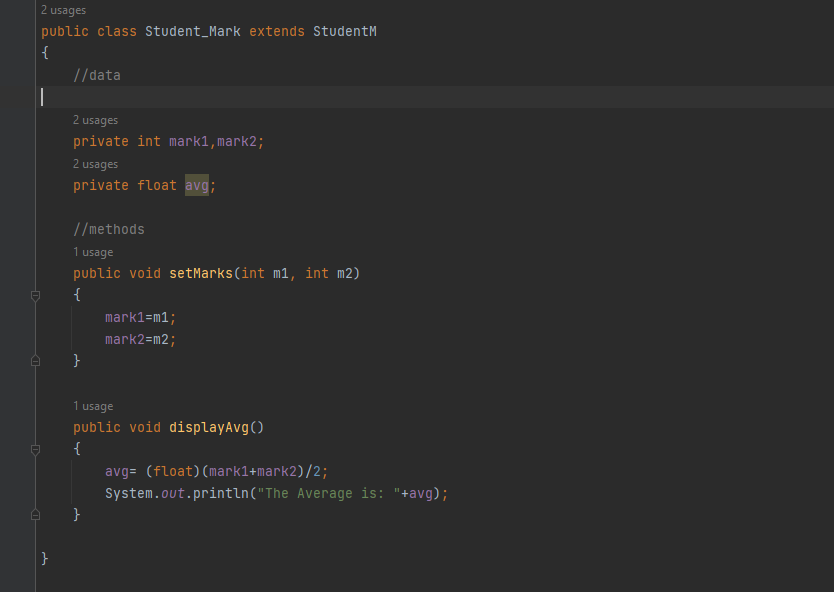
// Another Class which shows student Marks where it is called as a class Student\_Mark where it shows data for mark1, mark2, and method to find the average.

//Finally the object is calledd inside the main function – which is the important part of this inheritance where we able to access the parent data via the syntax using “ Student\_Mark m1 = new Student\_Mark()”

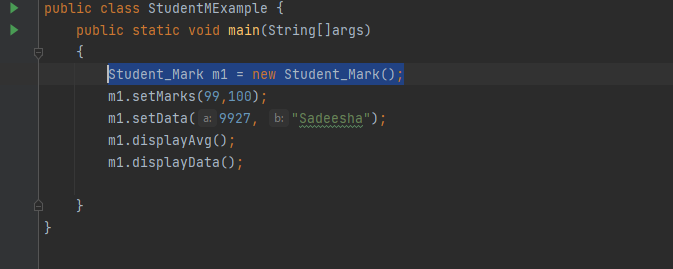
**StudentM Class**



**Student\_Mark Class**



**The Object Code – StudentMExample**



**2023.05.29**

Q - Write the difference between private vs public access modifiers

Private – data & methods can only be accessed within the class.

Public – data & methods can access by any class in the program.

Q – What is the implication of declaring a variable or method as protected.

Once you declare the base class variables as protected the immediate child class can access the protected variables

Example :

Public class A

{

Protected int a,b;

Public void setA(int a, int b)

{

this.a=a;

this.b=b;

}

}

Public class B extends A

{

Private int c;

Public void setB(int c)

{

This.c=c;

}

Public void display()

{

System.out.println(“a is”,+a);

System.out.println(“b is”,+b);

System.out.println(“c is”,+c);

}

}

**Main Class**

Public class AccessMod

{

Public static void main String([] args)

{

B obj1=ne,w B();

Obj1.setA(85,110);

Obj1.setB(50);

Obj.displayB();

}

**Constructers inside the child class**

There are two types of constructers which are with parameter and without parameter.

Example

Public class A

{

Private int a,b;

//parametrized constructer

Public A(int a, int b)

{

this.a=a;

this.b=b;

}

Public void display()

{

System.out.println(“a is”,+a);

System.out.println(“b is”,+b);

}

Public class B extends A

{

Private int c;

//parametrized constructer

Public B(int a , int b, int c)

{

Super(a,b);

this.c=c;

}

Public void displayB()

{

System.out.println(“c is”,+c);

}

}

**Important**

In the above program the child class constructer has three parameters it means the inherited class constructer pass values to the base class constructer in addition to its own variables.

Public class Accessmod {

Public static void main(String[] args)

{

B obj1=new B();

obj1.setA(10,20,30);

obj1.displayA();

obj1.displayB();

}

}

**Interfaces**

In java an interface contains constant variables and method prototypes (methods without body).

Example

Interface PaySalary

{

Final float allowance =5000.00f;

Void calcIncentive();

}

class A implements PaySalary

{

Public void calcIncentive()

{

}

}

The interface methods are implemented by the other classes.

The reason to use interfaces is as java doesn’t support multiple inheritance. Therefore, the method calls interfaces have been used.

The interface is used as an alternative to the multiple inheritance in java.

**Abstract Classes and Abstract Methods**

The Abstract class is a similar concept to the interface. But the difference is the abstract class contains methods without body and methods with implementation.

Example:

Abstract class Sample

{

Public void display()

{

}

abstract void calculate();

}

Class Test Extends Sample

{

Public void calculate()

{

}

}

**2023.06.05**

**Exercise**

Interface calPay

{

Final float allowance=12500.00f;

void calPay();

}

class A implement calcPay

{

Public void calPay()

{

//……..

}

}

Exercise No 2:

class A

{

}

class B extend A

{

}

Ex 3:

abstract class Shape

{

abstract void calArea();

public void display()

{………….}

}

**Exercise – Take Home**

1. Create the class Student with the following data and methods.

Data-Admission No, First name.

Methods- SetValues(int, String), display()

1. Another class Marks inherits from the student class. It has the following data and methods.

Data- module1, module2, module3

Methods- setMarks(int,int,int), displayAverage()

1. Create another class, with the main method to create an Object from the Marks class and call all 4 methods.

**Polymorphism**

The polymorphism means having many forms. It is divided into two types in methods. Which are as follows.

1. Method overloading – inside a class you include more than one method with the same method name, but with different number of parameters.

//method overloading

Class A

{

Public int claArea(int a)

{

Return a\*a;

}

Public int claArea(int a,int b)

{

Return a\*b;

}

}

1. Method overriding - Method overriding is related with the inheritance. In here same method name is use inside a base class and the child class with different method bodies.

//method overriding

Class A

{

Public void display()

{

//………..

}

}

Class B extend A

{

Public void display()

{

//………..

}

}

**2023.06.19 – Note Continue**

**Static Variables and Static Methods**

**Static Variables**

* The static variables are common across all the objects. It’s not varying from object to another object.
* We can access the static variables through the class name instead of accessing through the object.

**Static Methods**

* A Static Method can access the static variables and modify the value.
* Static methods can be call through the class name instead of calling through an object.

Public class Student {

int admNo;

String name;

static String institute="NSBM";

static int studentcount=0;

Public Student (int a, String n){

admNo=a;

name="n";

studentcount++;

System.out.println("Student Count is "+studentcount);

}

Public void display(){

System.out.println(admNo+""+name+""+institute);

}

}

**Object class**

Public static void main(String[] args) {

Student s1=new Student(1234,"Anne");

s1.display();

Student s2=new Student(7890,"James");

s2.display();

System.out.println("NSBM Green Uniwersity");

s2.display();

}

}

**Exception Handling**

* One of the simplest ways to handle an exception is to use try/catch block.
* The following example explain how we can handle dividing a number by zero exception.

Public class Studentobj2 {

Public static void main(String[] args) {

try{

int x=10,y=0,z;

z=x/y;

System.out.println("Z is "+z);

catch(Exception e){

System.out.println(e.getMessage());

}

}

* In the above program exception is a base class of all the exceptions.

1.Write a program to allow the user to input two numbers and divide first number by second number and display the answer. If the user enters the second number as zero, the program built generate and exception. By using try catch handle the exception.

import java.util.Scanner;

Public class TextExceptions {

Public static void main(String[] args) {

int no1,no2,ans;

Scanner sc=new Scanner (System.in);

try{

System.out.println("Enter First Number ");

no1=sc.nextInt();

System.out.println("Enter Second Number ");

no2=sc.nextInt();

ans=no2/no1;

System.out.println("Answer is "+ans);

}

catch(Exception e){

System.out.println("Number divide by zero error ");

}

}

}

To try catch clause it means the final block will execute weather and exception is generated or not.

Public class TextExceptions {

Public static void main(String[] args) {

int no1=10,no2=2,ans;

//Array in java

int arr[]=new int[5];

try{

ans = no1/no2;

System.out.println("Answer is "+ans);

arr[4]=25;

System.out.println("Last elemnt is "+arr[4]);

}

catch(ArithmeticException e){

System.out.println("Number divided by zero error ");

}

catch(ArrayIndexOutOfBoundsException e){

System.out.println("Array inbox errro ");

}

finally{

System.out.println("end");

}

}

}

**Throw/Throws**

This is another method of handling exceptions. This is mostly use with check exception.

*Age class*

Public class Age {

Public void checkAge(int a)throws ArithmeticException{

if(a<18){

throw new ArithmeticException("You are not allowed to login");

}

else{

System.out.println("Welcome !");

}

}

}

*TextExceptions Object class*

Public class TextExceptions {

Public static void main(String[] args) {

try{

Age a=new Age();

a.checkAge(7);

}

catch (ArithmeticException e){

System.out.println(e.getMessage());

}

}

}

*TextExceptions Object class*

Public class TextExceptions {

Public static void main(String[] args) {

Person p=new Person(21,"Anne");

//when you display the display the object name, it will call the totring() method

System.out.println(p);

}

}

**User Defined Exception**

**Multithreading**

2023.07.11

Multithreading is a form of multitasking. Which is basically executing several tasks at the same time.

Multithreading is efficient compared to multiprocessing.

There are two methods or approaches to create a thread in java.

1. Extends from the thread class. (Inheritance)
2. Implements the runnable interface.
3. Extends the Thread Class

Example 1

// for more knowledge about the code

The given code is an example of multithreading in Java. It includes two classes: `ThreadExample` and `ThreadA`.

The `ThreadExample` class has a `main` method that serves as the entry point of the program. Within the `main` method, two instances of the `ThreadA` class are created: `t1` and `t2`.

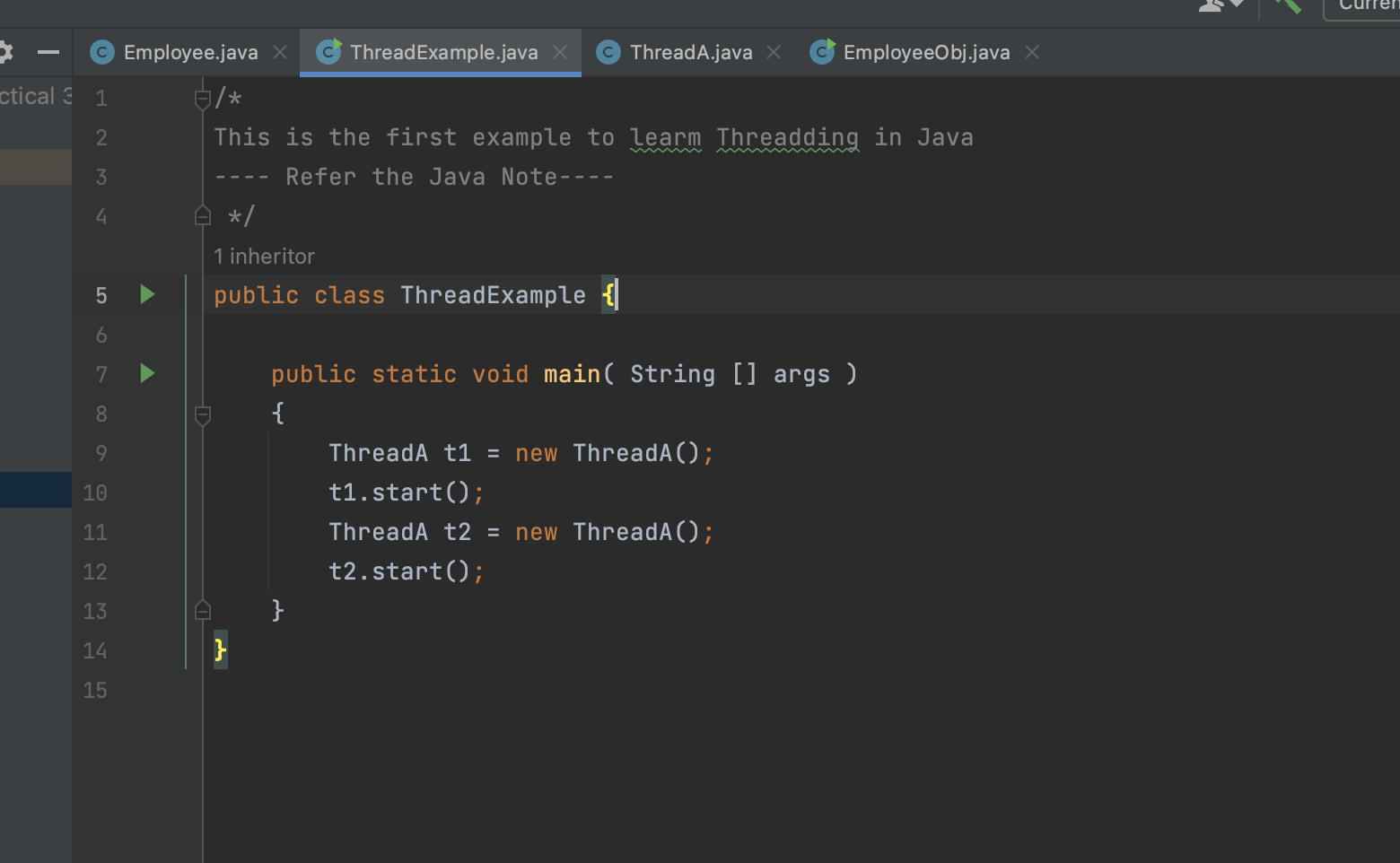
The `ThreadA` class extends the `ThreadExample` class, which seems unnecessary in this context. Typically, when creating a separate thread, there is no need for the thread class to extend another class.

Inside the `run` method of `ThreadA`, there is a loop that prints the values of `x` from 1 to 5.

To correct the code, you can remove the inheritance relationship between `ThreadA` and `ThreadExample` since it is unnecessary. Additionally, it's good practice to override the `run` method of the `Thread` class directly instead of extending another class.

In this updated code, the `ThreadA` class extends the `Thread` class directly and overrides the `run` method. Inside the `run` method, the loop prints the values of `x` from 1 to 5.

By creating multiple instances of `ThreadA` and starting them, you can run multiple threads concurrently, and each thread will execute the code inside its `run` method independently.



The ThreadA Code

A screen shot of a computer program

Description automatically generated

JAVA THREAD METHODS

Start()

Run ()

Sleep ()

Wait ()

Yield ()

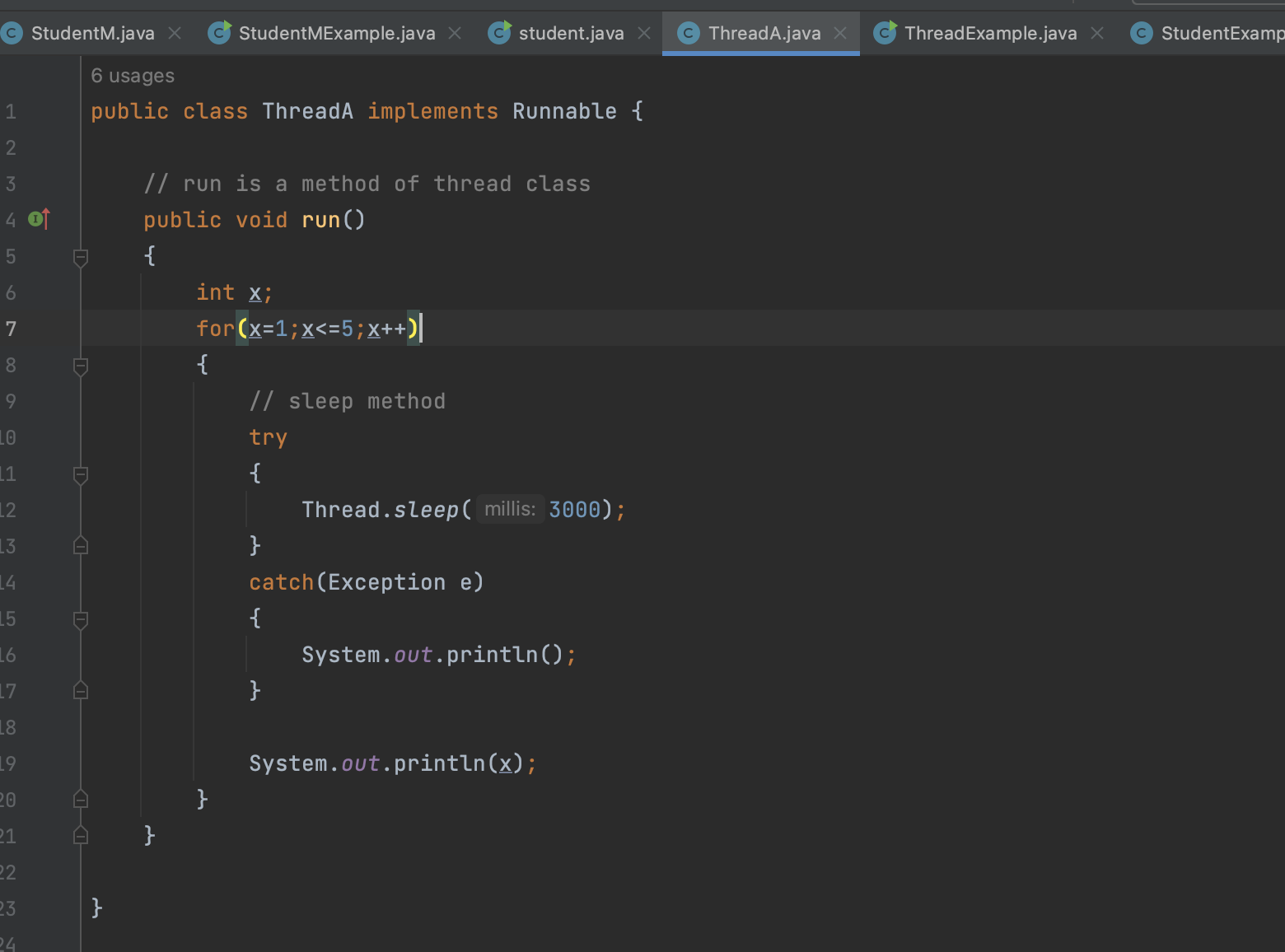
Join ()

How the sleep () method works?  
The java sleep method temporarily stops it’s execution for a given number of milliseconds.

The sleep method is a static method, and it generates check exemption.

1. Implement the Runnable Interface

There is a simple change has taken place when using the runnable interface for threading which is that “public class ThreadA implements Runnable. “ which can be seen in the below image.



The main Class – There is a small difference when creating objects than the first method which can be seen in the below image. ssssss

