OBJECT, CLASS, USER INPUTS AND MATH FUNCTIONS

In object-oriented programming (OOP), a class and an object are fundamental concepts that facilitate the modeling of real-world entities and their interactions. Let's delve deeper into each:

Class:

- **Definition**: A class is a blueprint or template for creating objects. It defines a data type, describing the attributes (data members or fields) and behaviors (methods or functions) that objects created from the class will have.

- **Attributes**: The attributes represent the properties or characteristics of the objects. These are defined as variables within the class.

- **Methods**: The methods represent the actions or behaviors that the objects can perform. These are defined as functions within the class.

Example: Java Code

```
public class Car {
    // Attributes
    String model;
    int year;

    // Method
    void start() {
        System.out.println("The car is starting.");
    }
}
```

Object:

- **Definition**: An object is an instance of a class. It is a runtime entity that represents a specific occurrence of the class, with its own set of attribute values. Objects encapsulate data and behavior, and multiple objects can be created from the same class.

- **Creation**: Objects are created using the `new` keyword followed by the class constructor. The constructor initializes the object with default or provided values.

Example: Java Code

```
public class Main {
    public static void main(String[] args) {
        // Creating an object of the Car class
        Car myCar = new Car();

        // Accessing attributes and invoking methods of the object
        myCar.model = "Toyota";
        myCar.year = 2022;
        myCar.start();
    }
}
```

In the example above, `myCar` is an object of the `Car` class. It has attributes (`model` and `year`) and a method (`start`). Multiple objects can be created from the `Car` class, each representing a different car with its unique characteristics and behaviors.

In summary, a class is a blueprint that defines the structure and behavior of objects, while an object is an instance of a class, representing a specific entity with its own set of attribute values and behaviors. OOP allows for the creation of modular and reusable code by organizing it around the concepts of classes and objects.

Math Class in Java

Math class file is included for the definitions of math functions listed below. It is written as java.lang.Math

Maths Functions
sin(n)
cos(n)
tan(n)
sinh(n)
cosh(n)
tanh(n)
pow(nmb,pwr)
sqrt(n)

In Java, mathematical functions are available in the `java.lang.Math` class. This class provides a set of static methods for performing common mathematical operations. Here are some examples of mathematical functions in Java:

Trigonometric Functions:

1. sin(n):

Java Syntax: double result = Math.sin(Math.toRadians(n));

2. cos(n):

Java Syntax: double result = Math.cos(Math.toRadians(n));

3. tan(n):

Java Syntax: double result = Math.tan(Math.toRadians(n));

4. sinh(n):

Java Syntax: double result = Math.sinh(n);

5. cosh(n):

Java Syntax: double result = Math.cosh(n);

6. tanh(n):

Java Syntax: double result = Math.tanh(n);

Exponential and Power Functions:

7. pow(nmb, pwr):

Java Syntax: double result = Math.pow(nmb, pwr);

Square Root:

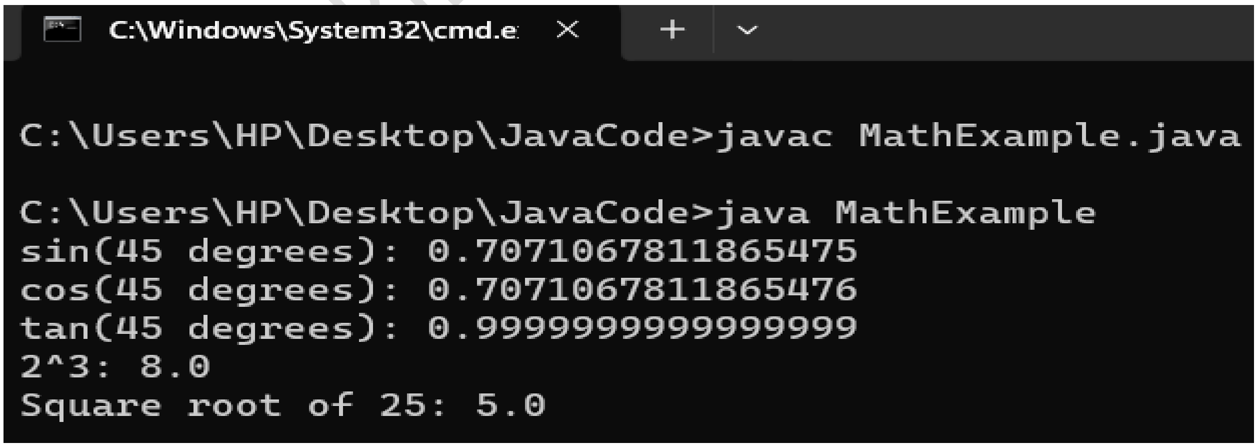
8. sqrt(n):

Java Syntax: double result = Math.sqrt(n);

Here's an example of using some of these functions:

```
import java.lang.Math;
public class MathExample {
    public static void main(String[] args) {
        double angleInDegrees = 45.0;
        double angleInRadians = Math.toRadians(angleInDegrees);
        double sinValue = Math.sin(angleInRadians);
        double cosValue = Math.cos(angleInRadians);
        double tanValue = Math.tan(angleInRadians);
        double powResult = Math.pow(2, 3);
        double sqrtResult = Math.sqrt(25);

        System.out.println("sin(45 degrees): " + sinValue);
        System.out.println("cos(45 degrees): " + cosValue);
        System.out.println("tan(45 degrees): " + tanValue);
        System.out.println("2^3: " + powResult);
        System.out.println("Square root of 25: " + sqrtResult);
    }
}
```



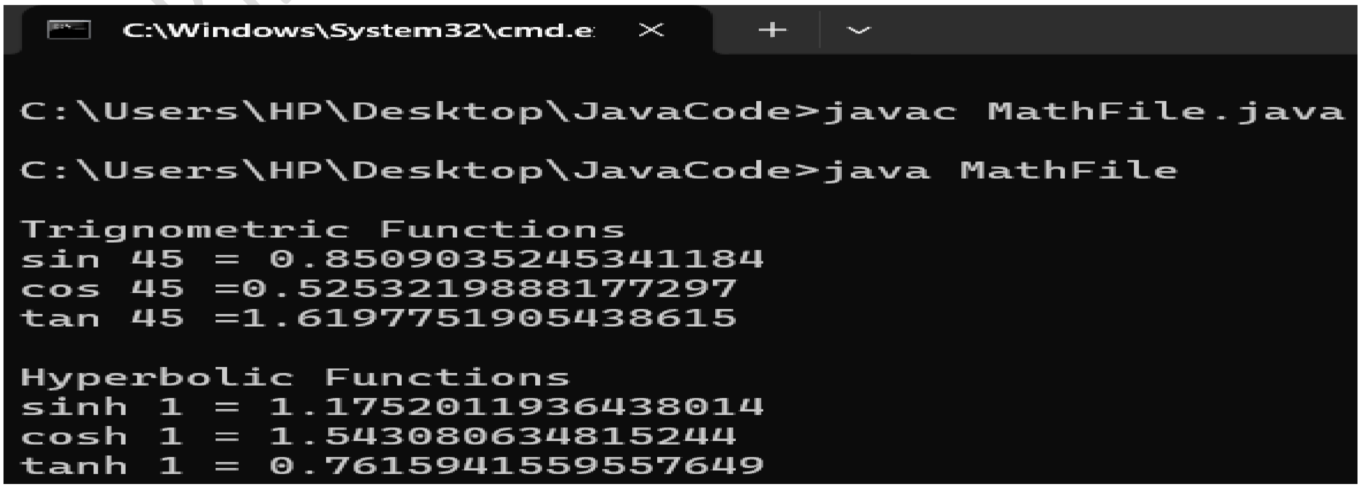
```
C:\Users\HP\Desktop\JavaCode>javac MathExample.java

C:\Users\HP\Desktop\JavaCode>java MathExample
sin(45 degrees): 0.7071067811865475
cos(45 degrees): 0.7071067811865476
tan(45 degrees): 0.9999999999999999
2^3: 8.0
Square root of 25: 5.0
```

In this example, trigonometric functions are used to calculate the sine, cosine, and tangent of an angle. Additionally, the `pow` method is used for exponentiation, and the `sqrt` method is used for finding the square root. Remember to convert angles to radians when working with trigonometric functions.

Another Example:

```
import java.lang.Math;
class MathFile {
public static void main(String[] args) {
double a=45,b=1,sn,cs,tn,snh,csh,tnh;
sn=Math.sin(a);
cs=Math.cos(a);
tn=Math.tan(a);
snh=Math.sinh(b);
csh=Math.cosh(b);
tnh=Math.tanh(b);
System.out.println("\nTrignometric Functions");
System.out.println("sin 45 = " + sn);
System.out.println("cos 45 =" + cs);
System.out.println("tan 45 =" + tn);
System.out.println("\nHyperbolic Functions");
System.out.println("sinh 1 = " + snh);
System.out.println("cosh 1 = " + csh);
System.out.println("tanh 1 = " + tnh);
  }
}
```



```
C:\Users\HP\Desktop\JavaCode>javac MathFile.java

C:\Users\HP\Desktop\JavaCode>java MathFile

Trignometric Functions
sin 45 = 0.8509035245341184
cos 45 =0.5253219888177297
tan 45 =1.6197751905438615

Hyperbolic Functions
sinh 1 = 1.1752011936438014
cosh 1 = 1.543080634815244
tanh 1 = 0.7615941559557649
```

Methods to Take Input in Java:

There are two ways by which we can take Java input from the user or from a file

1. **BufferedReader Class**
2. **Scanner Class**

1. Using **BufferedReader Class** for String / Int / Float Input in Java

It is a simple class that is used to read a sequence of characters. It has a simple function that reads a character another read which reads, an array of characters, and a readLine() function which reads a line.

InputStreamReader() is a function that converts the input stream of bytes into a stream of characters so that it can be read as BufferedReader expects a stream of characters. BufferedReader can throw checked Exceptions.

Below is the implementation of the **BufferedReader** Class approach:

```
import java.io.*;

class BufferReader {

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

// Creating BufferedReader Object, InputStreamReader converts bytes to stream of character

        BufferedReader bfn = new BufferedReader(new InputStreamReader(System.in));


                System.out.print("Enter String : ");

                String str = bfn.readLine();                    // String reading internally

                System.out.print("Entered Integer : ");

                int it = Integer.parseInt(bfn.readLine());       // Integer reading internally

                System.out.print("Entered Float : ");

                float ft = Float.parseFloat(bfn.readLine());     // Float reading internally

                System.out.println("Entered String : " + str);   // Printing String

                System.out.println("Entered Integer : " + it);   // Printing Integer

                System.out.println("Entered Integer : " + ft);   // Printing Float

        }
}
```

2. Using **Scanner Class** for Taking Input In Java

It is an advanced version of BufferedReader which was added in later versions of Java. The scanner can read formatted input. It has different functions for different types of data types.

- The scanner is much easier to read as we don't have to write throws as there is no exception thrown by it.
- It was added in later versions of Java
- It contains predefined functions to read an Integer, Character, and other data types as well.

Syntax of Scanner class

Scanner scn = new Scanner(System.in);

Importing Scanner Class

To use the Scanner we need to import the Scanner class from the util package as

import java.util.**Scanner**;

Inbuilt Scanner functions are as follows:

Integer: nextInt()

Float: nextFloat()

String : next() and nextLine()

Hence, in the case of Integer and String in Scanner, we don't require parsing as we did require in BufferedReader.

Java Packages & API

A package in Java is used to group related classes. Think of it as a folder in a file directory. We use packages to avoid name conflicts, and to write a better maintainable code. Packages are divided into two categories:

Built-in Packages (packages from the Java API)

User-defined Packages (create your own packages)

To use the Scanner class, create an object of the class and use any of the available methods found in the Scanner class documentation. In our example, we will use the nextLine() method, which is used to read Strings:

Example: Java Program to show how to take input from user using Scanner Class

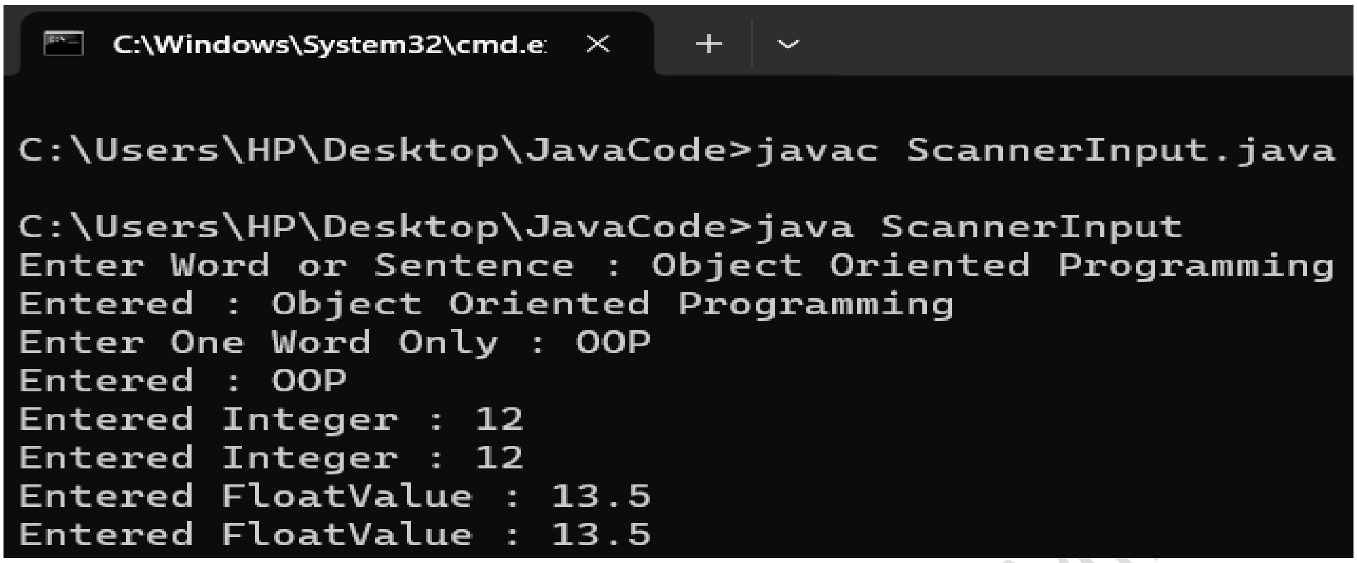
```
import java.util.Scanner;
class ScannerInput {
    public static void main(String[] args)
    {
        Scanner takeInp = new Scanner(System.in);          // Scanner definition

        // input is a String ( complete Sentence ) read by nextLine()function
        System.out.print("Enter Word or Sentence : ");
        String str1 = takeInp.nextLine();
        System.out.println("Entered : " + str1);           // print String

        // input is a string ( one word ) read by next() function
        System.out.print("Enter One Word Only : ");
        String str2 = takeInp.next();
        System.out.println("Entered : " + str2);           // print String

        // input is an Integer read by nextInt() function
        System.out.print("Entered Integer : ");
        int x = takeInp.nextInt();
        System.out.println("Entered Integer : " + x);      // print integer

        // input is a floatingValue read by nextFloat() function
        System.out.print("Entered FloatValue : ");
        float f = takeInp.nextFloat();
        System.out.println("Entered FloatValue : " + f);   // print floating value
    }
}
```



```
C:\Users\HP\Desktop\JavaCode>javac ScannerInput.java

C:\Users\HP\Desktop\JavaCode>java ScannerInput
Enter Word or Sentence : Object Oriented Programming
Entered : Object Oriented Programming
Enter One Word Only : OOP
Entered : OOP
Entered Integer : 12
Entered Integer : 12
Entered FloatValue : 13.5
Entered FloatValue : 13.5
```

Differences Between BufferedReader and Scanner

- BufferedReader is a very basic way to read the input generally used to read the stream of characters. It gives an edge over Scanner as it is faster than Scanner because Scanner does lots of post-processing for parsing the input; as seen in nextInt(), nextFloat()
- BufferedReader is more flexible as we can specify the size of stream input to be read. (In general, it is there that BufferedReader reads larger input than Scanner)
- These two factors come into play when we are reading larger input. In general, the Scanner Class serves the input.
- BufferedReader is preferred as it is synchronized. While dealing with multiple threads it is preferred.
- For decent input, and easy readability. The Scanner is preferred over BufferedReader.

Input Types

To read other types, look at the Examples below:

nextBoolean()	Reads a boolean value from the user
nextByte()	Reads a byte value from the user
nextDouble()	Reads a double value from the user
nextFloat()	Reads a float value from the user
nextInt()	Reads a int value from the user
nextLine()	Reads a String value from the user
nextLong()	Reads a long value from the user
nextShort()	Reads a short value from the user