Object Oriented Programing 1

Lecture 4: Writing First Java Program, Data type and Identifier & Variable in Java

By

Elubu Joseph

MSci.IS

Email: josebulinda@gmail.com

or

jose@kumiuniversity.ac.ug

Agenda

1. Identifiers,
2. Writing java program,
3. Data types and
4. Variables in Java

Identifiers in Java

Identifiers in Java

Names given to classes, methods, interfaces, constants and variables by the programmer are called **Identifier**.

Identifier must follow some rules. Here are the rules:

1. All identifiers must start with either a letter(a to z or A to Z) or currency character($) or an underscore but **not** a number.
2. After the first character, an identifier can have any combination of characters.
3. Java **keywords** cannot be used as an identifier.
4. Identifiers in Java are case sensitive, **Teach** and **teacH** are two different identifiers.
5. No special characters, such as semicolon, period, whitespaces, slash or comma are permitted to be used in or as Identifier.

Java Keywords

Keywords are reserved words that are exclusively used by java programing language to facilitate program development. You cannot use any of the keywords as an identifiers in your programs.

List of keywords.

abstract	continue	for	new	switch
assert***	default	goto*	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum****	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp**	volatile
const*	float	native	super	while

Java Keywords+

Note! The keywords **const** and **goto** are reserved, even though they are not currently used. **true**, **false**, and **null** might seem like keywords, but they are actually literals; you cannot use them as identifiers in your programs.

Writing Java Program

First Program

A program is set of instructions written to accomplish a given task or tasks. It consists of various lines of code. Since Java is a Object-oriented language so it require to write a code inside a class. Let us look at a simple java program that prints Hello Programmer on the screen.

```
class Fpro{
public static void main(String[] args) {
System.out.println (“Hello Programmer");
 }
}
```

Output

Hello Programmer

Major parts of a program

Lets understand what the above program consists of and its key points.

class : class keyword is used to declare classes in Java. Naming a class

public : It is an access specifier. Public means this function is visible to all.

static : static is a keyword used to make a method static. To execute a static method, you do not have to create an Object of the class.

The **main()** is the method head which is called by JVM, without creating any object for class. **main** is the name of the method.

Major parts of a program+

main : main() method is the most important method in a Java program. This is the method which is executed, hence all the logic must be inside the main() method. If a java class is not having a main() method, it causes compilation error, and the program will not run on its own, unless it is run by another class.

void : is the return type, meaning this method will not return anything.

String[] args : This represents an array whose type is **String** and name is **args**. We will discuss more about array in Java Array lecture.

System.out.println() : This is java output stream method used to print anything on the console like *printf* in C language.

Ways to write a Java Program

The Following are some of the ways in which a Java program can be written:

Changing the sequence of the modifiers and methods is accepted by Java.

Syntax: static public void main(String args[])

Example:

```
class Fpro {
static public void main(String as[]) {
System.out.println (“Hello friends we are programming in java");
}
}
```

Output

Hello friends we are programming in java

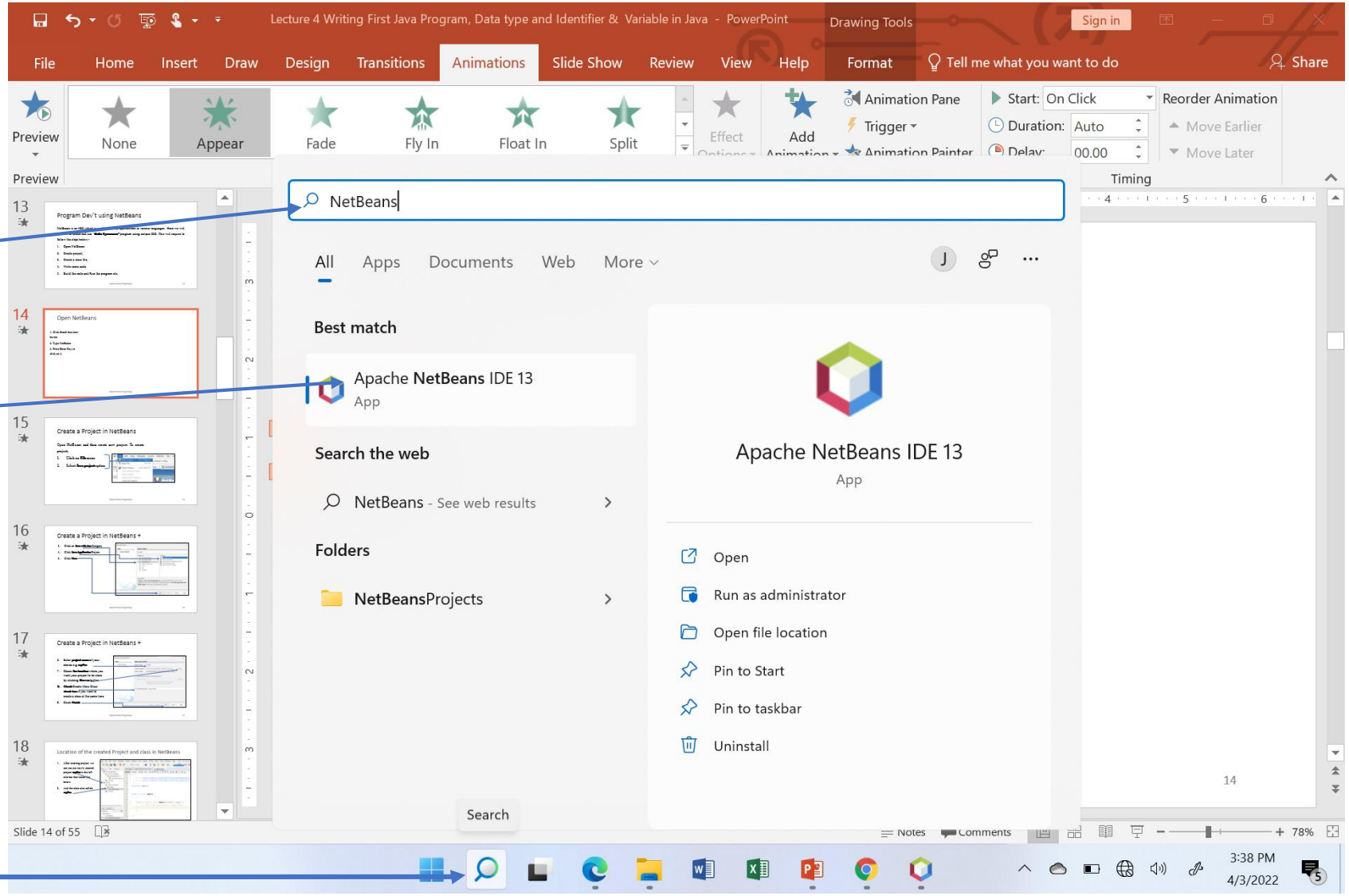
Program Dev’t using NetBeans

NetBeans is an IDE which is used to develop applications in various languages. Here we will see how to create and run ”**Hello Pgorammer”** program using eclipse IDE. This will require to follow the steps below;-

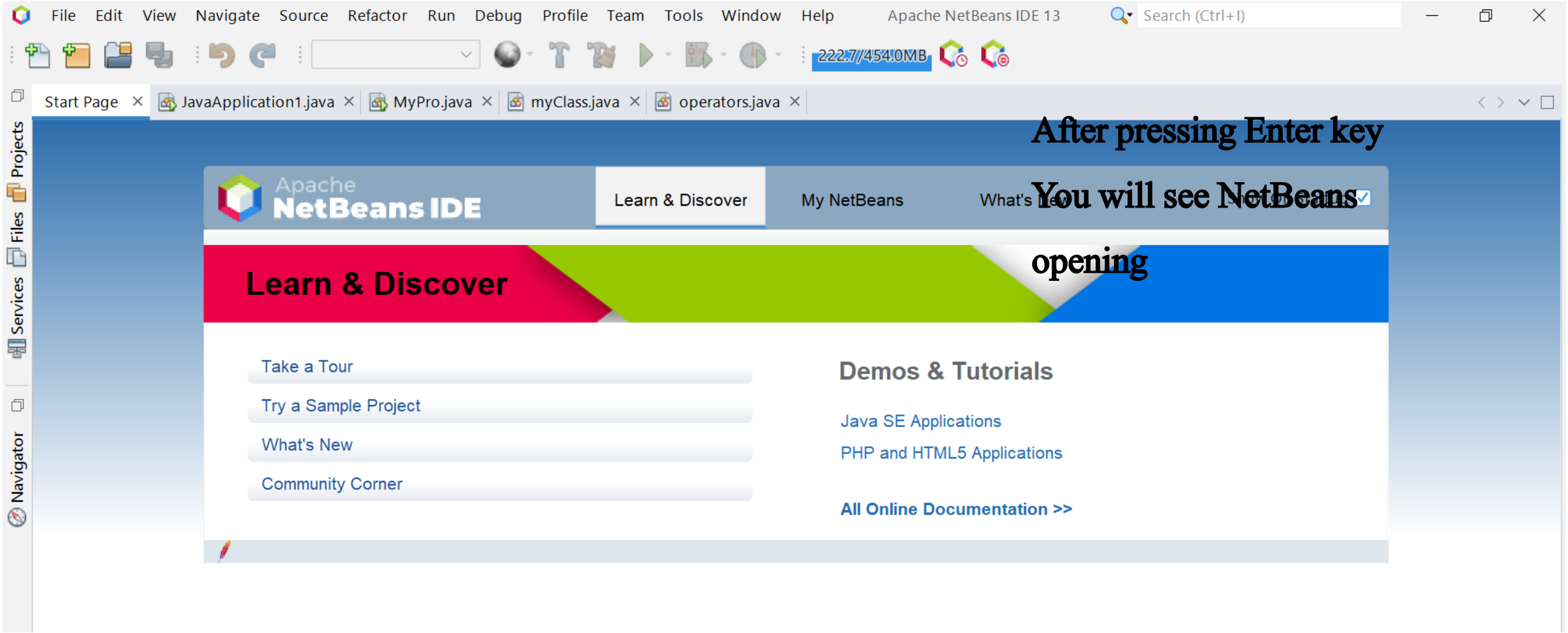
1. Open NetBeans
2. Create project,
3. Create a class file,
4. Write some code
5. Build the code and Run the program etc.

Open NetBeans

1. Click **Search box/start button**
2. Type **NetBeans**
3. Press Enter Key or click on it



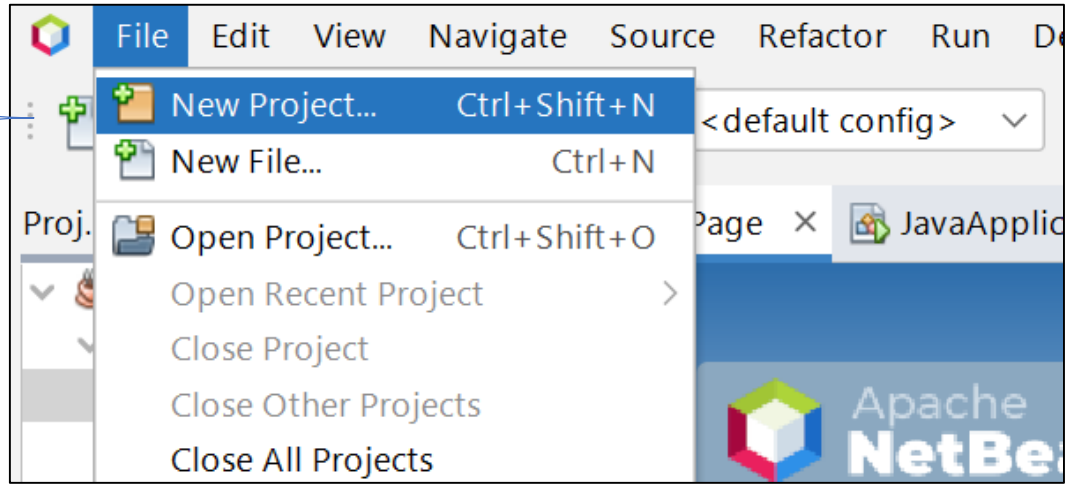
NetBeans IDE



Create a Project in NetBeans

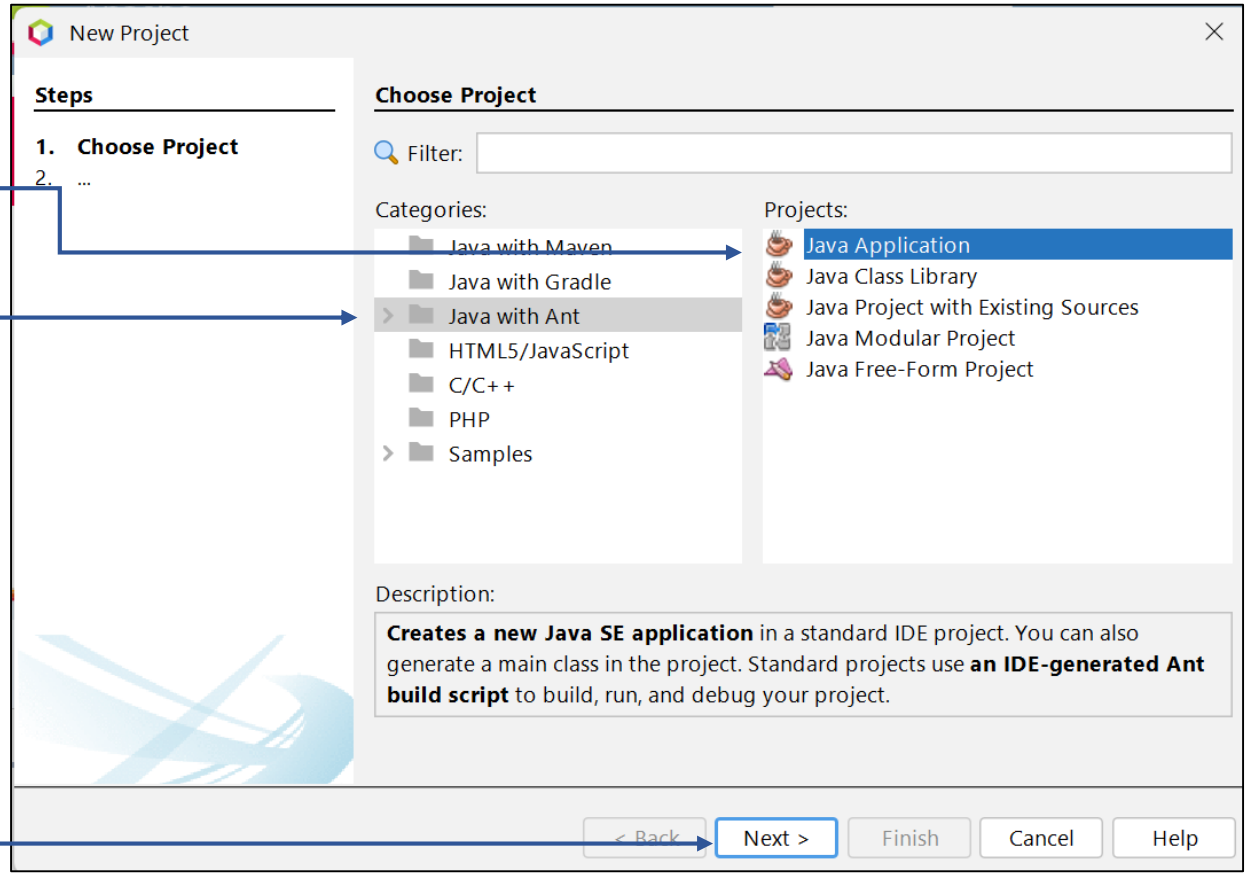
Having opened NetBeans you can now create new project. To create project,

1. Click on **File** menu
2. Select **New Project** option.



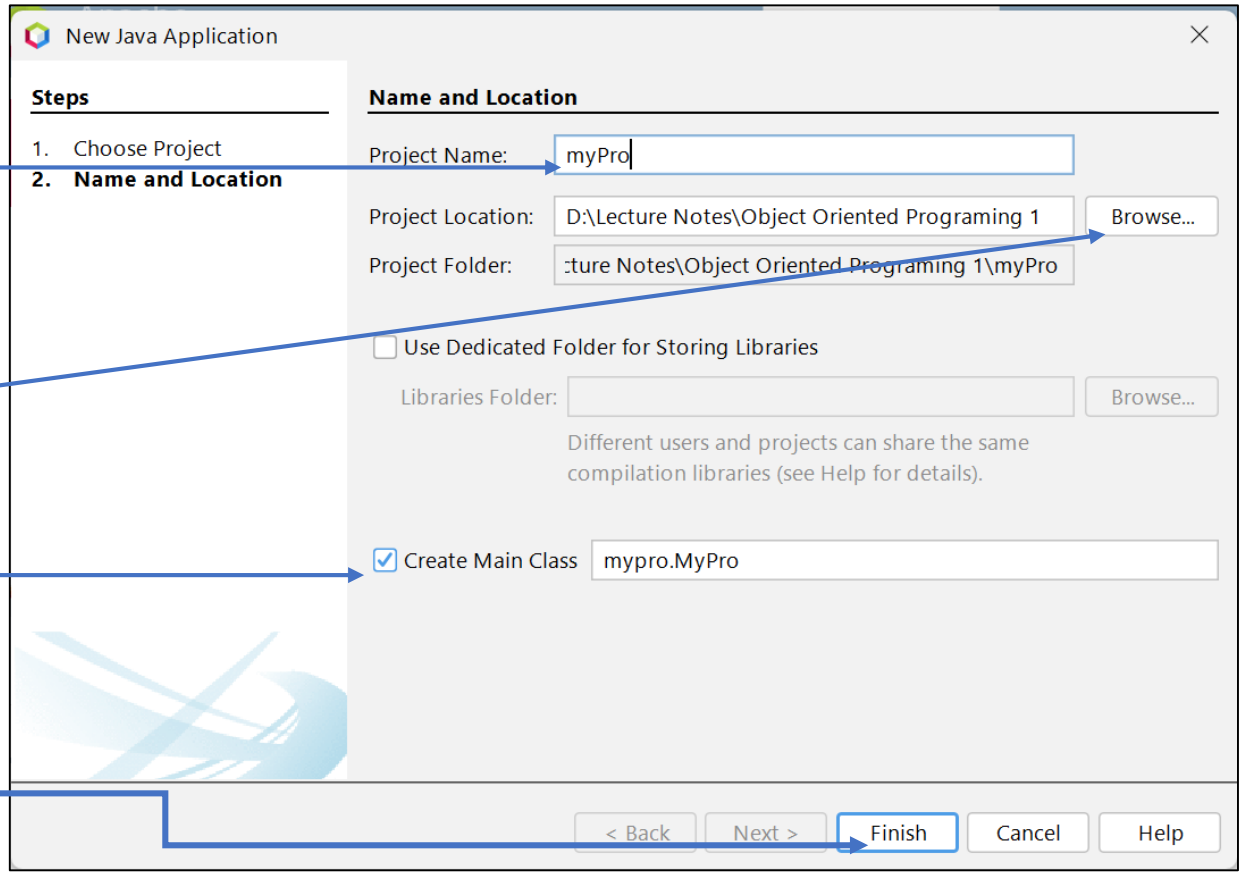
Create a Project in NetBeans +

3. Click on **Java with Ant** Category
4. Click **Java Application** Project
5. Click **Next**



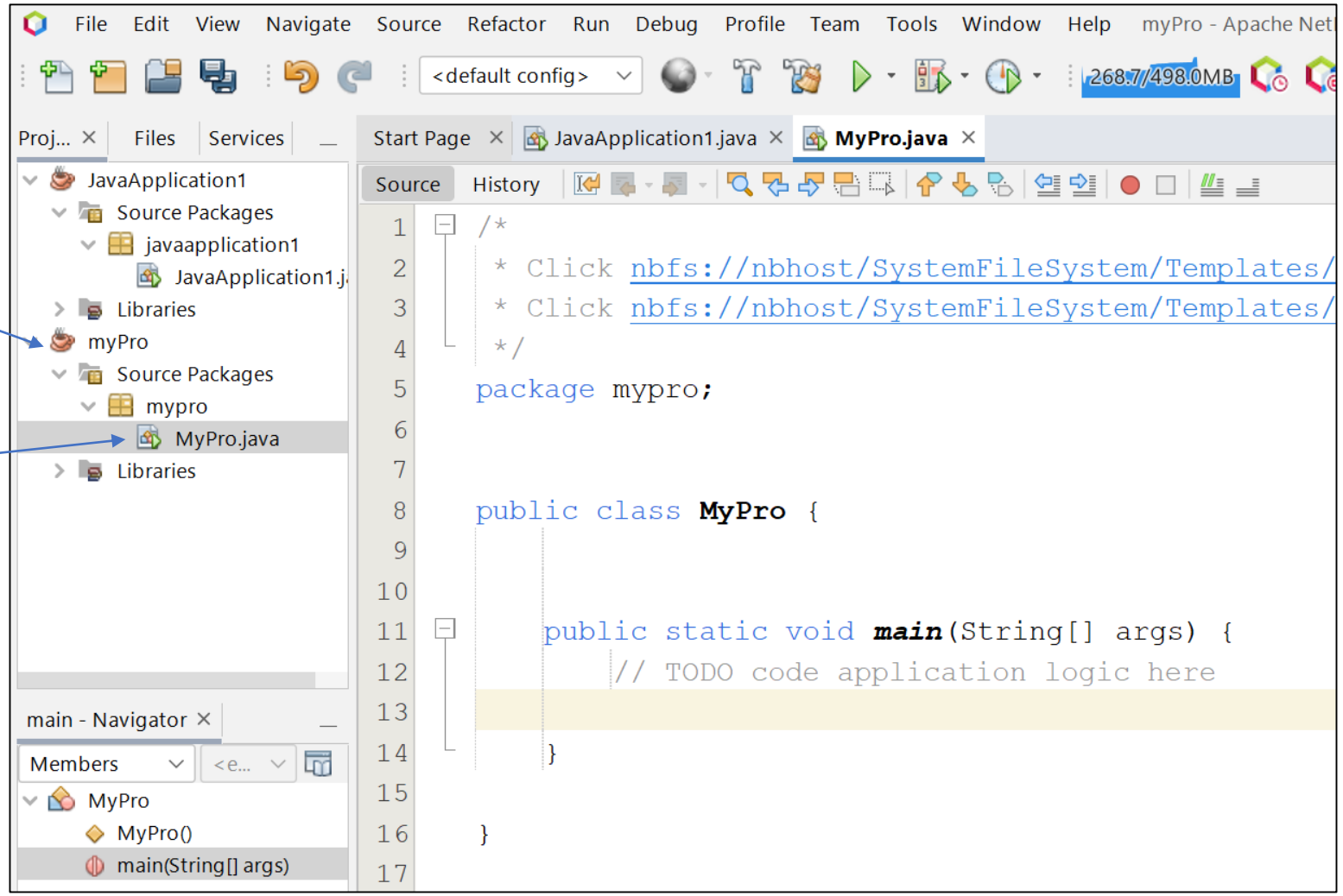
Create a Project in NetBeans +

6. Enter **project name** of your choice e.g **myPro**
7. Chose **the location** where you want your project to be store by clicking **Browse** button
8. Check **Create Main Class check box** if you want to create a class at the same time
9. Click **Finish**



Location of the created Project and class in NetBeans

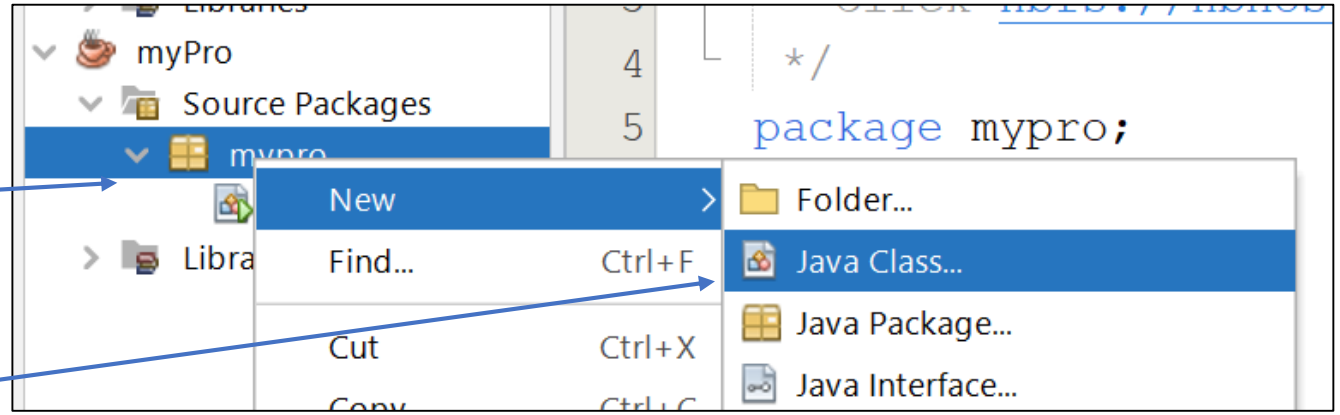
1. After creating a project, we can see our newly created project **myPro** in the left side bar that looks like below.
2. And the class also called **myPro**



Create Java Class

Now create Java class file by;-

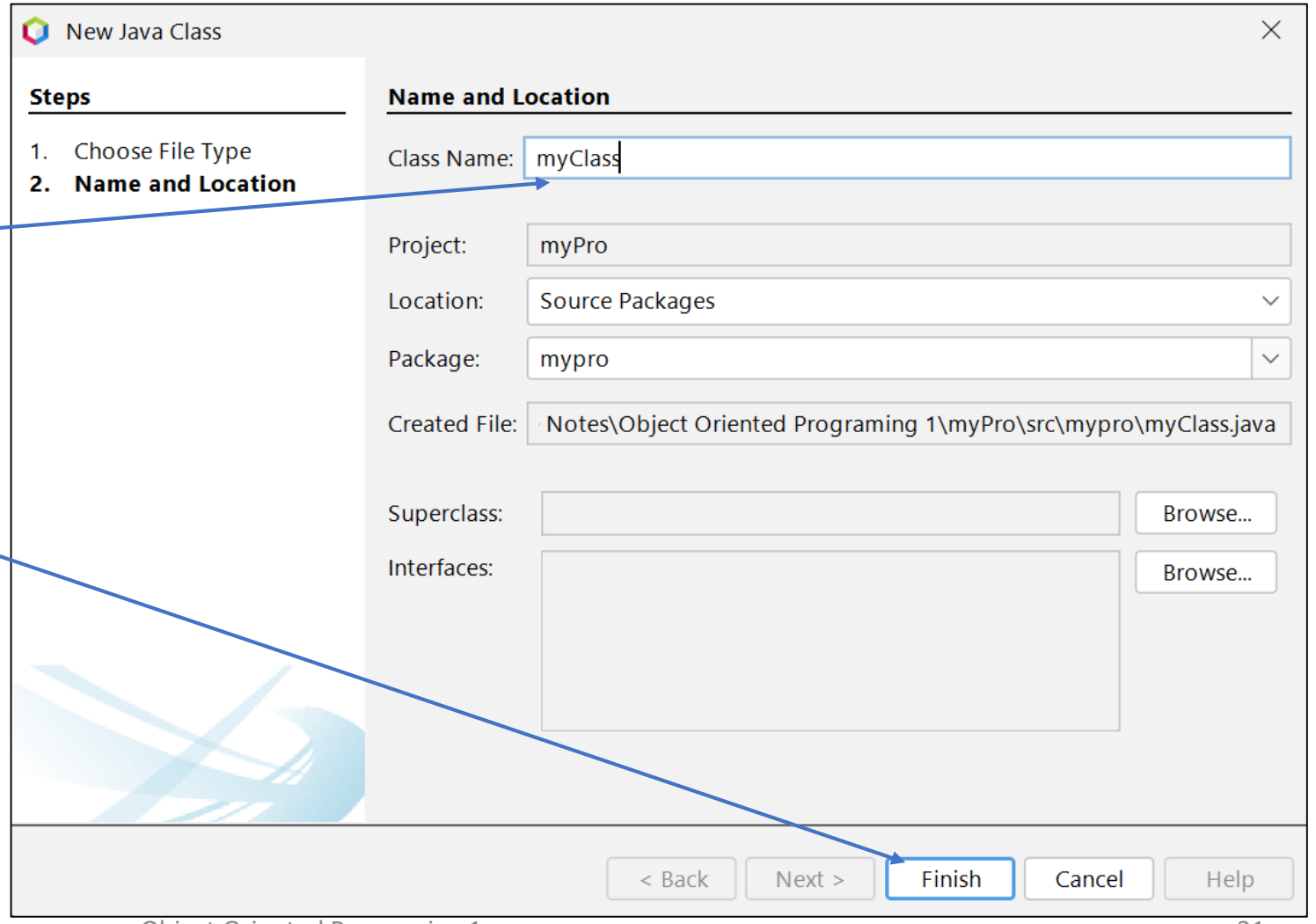
1. **Right clicking** on the **project or package name** and
2. **Point at New** then Select **Java class** file option. It will open a window to ask for class name,



Create Java Class +

Now create Java class file by;-

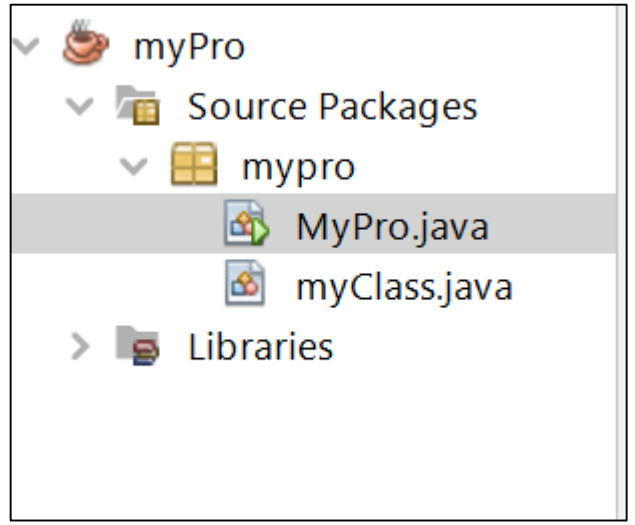
3. Provide the class name **e.g. myClass** and
4. Click on **Finish** button.



Enter Code into a class

Now we need to write just a print statement, to print **Hello Programmer** message in MyPro.java.

Note! By now you should be seeing two classes created that is myPro.java and myClass.java, one having main() method and the other not having main().



MyPro.java has **main()** method while myClass.java dose not have **main()** method

```
8   public class MyPro {
9
10      public static void main(String[] args) {
11          // TODO code application logic here
12
13      }
14
15  }
```

```
7   public class myClass {
8
9   }
10
```

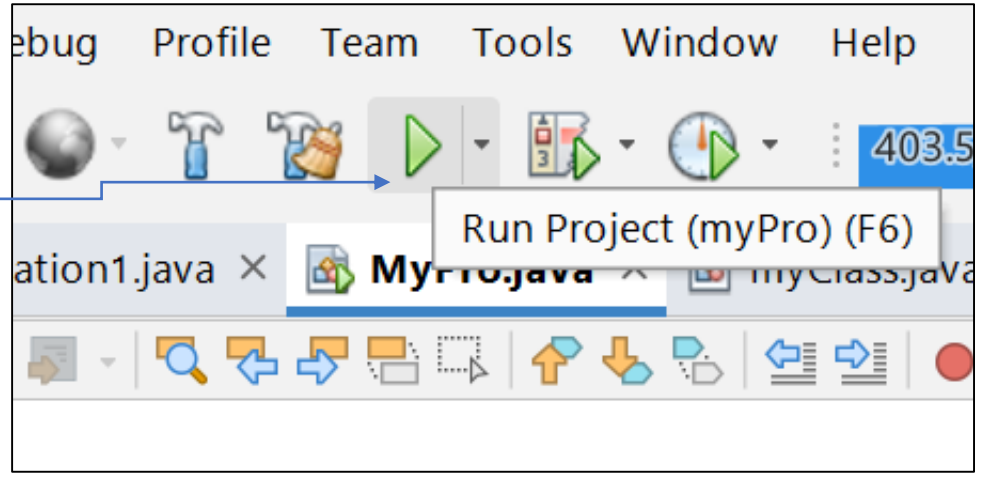
Enter Code into a class (MyClass.java)

Enter System.out.println("Hello Programmer"); inside the main(){ } body.

```
8   public class MyPro {
9
10      public static void main(String[] args) {
11
12          System.out.println("Hello Programmer");
13
14      }
15  }
```

Run The Program

Now run the program by Clicking on the **Run Project(myPro)**



How much can we do?

With the a simple program that we ran above, we can create and build large scale of applications.

If you are a beginner or not familiar with the NetBeans IDE don’t worry you will soon catch up just follow the above steps to create the program.

After compiling/Building you will try to run the byte code(*.class file), the following steps are performed at runtime:-

Now let us see What happens at Runtime

After writing your Java program, when you build it, The Compiler will perform some compilation operation on your source code into bytecode.

Once it is compiled successfully, bytecode (*.class) file is generated by the compiler.

Java Program transformation Process +

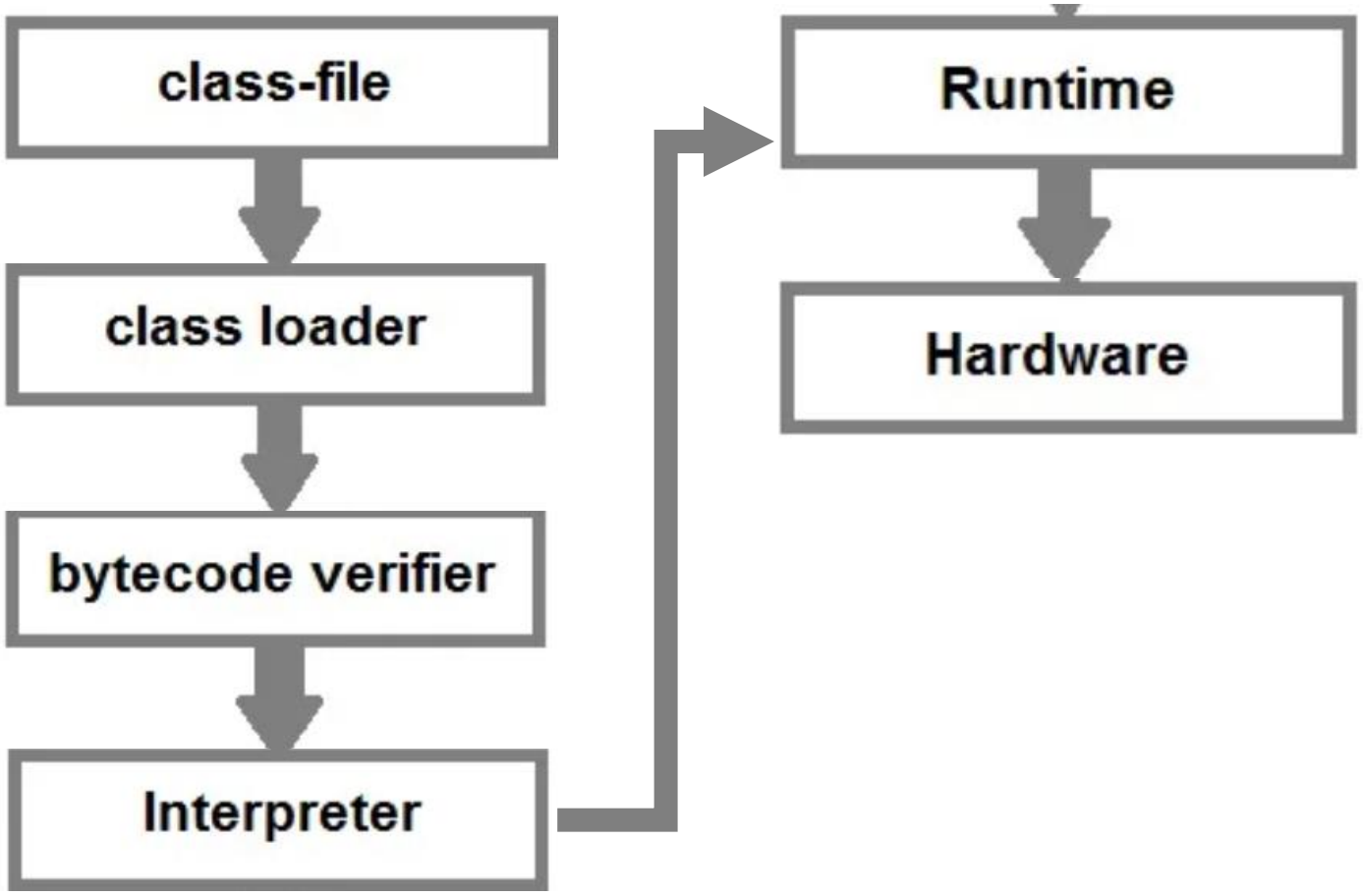
Class loader loads the java class to the JVM - Java Virtual machine.

Byte Code verifier checks the code fragments for illegal codes that can violate access right to the object.

Interpreter reads the byte code stream and then executes the instructions, step by step.

The end results of this process is the output, which is sent to the output device.

Java Program transformation Process



Variable in Java

Variable in Java

This section will help us learn about variables, various types of variables along with example codes to understand Java variables.

What is a Variable?

Variable is the name of memory location in a storage device.

When we want to store any information, we store it in an address of the computer. Instead of remembering the complex address where we have stored our information, we name that address. The naming of an address is known as variable declaration.

In other words, variable is a name which is used to store a value of any type during program execution. To declare the variable in Java, we can use following syntax

Variable Declaration

Is the act of naming the variable alongside its datatype.

```
datatype variableName;
```

Here, **datatype** refers to type of variable or kind of data a given variable is going to store, which can be anything like: **int, float** etc. and **variableName** can be anything like: **empId, amount, price** etc.

When we declare/name variables, we have to take note of the datatype followed by the name.

For example

int total; String name; double currency;

i.e **total**, **name**, and **currency** will store integer, string and double values respectively

Types of variables in Java

Java Programming language defines mainly three kind of variables which include:-

1. Instance Variables
2. Static Variables (Class Variables)
3. Local Variables

Instance variables in Java

are variables that are declared inside a class but outside any method, constructor or block. Instance variable are also variable of object commonly known as field or property.

They are referred as object variable. Each object has its own copy of each variable and thus, it doesn't effect the instance variable if one object changes the value of the variable.

- Here **name** and **age** are instance variable of Student class.

```
class Student{
String name;
int age;
}
```

Static variables in Java

are class variables declared with **static** keyword. Static variables are initialized only once. They are also used in declaring constants along with final keyword.

```
class Student {
String name;
int age;
static int countryCode=1101;
}
```

Here **countyCode** is a static variable. Each object of Student class will share countryCode property.

More points on static variable

static variable are also known as class variable.

1. static means to remain constant, meaning it will be constant for all the instances created for that class.
2. static variable need not to be called from object. It is called by *classname.static_variable_name*

Note: A static variable can never be defined inside a method i.e it can never be a local variable.

Example

Suppose you make 2 objects of class Student and you change the value of static variable from one object.

```
class Student{

static int id = 101;

void change(){
System.out.println(id);
}
}
```

Now when you print it from other object, it will display the **changed** value. This is because it was declared static i.e it is constant for every object created.

```
class statcStudent{

public static void main(String args[]){
  Student o1 = new Student();
  Student o2 = new Student();
  o1.change();

   Student.id = 45;
   o2.change();

}
}
```

Output: 101

Output: 45

Example

```
class Student{
static int id = 101;
void change(){
System.out.println(id);
}
}
public class department{
public static void main(String[] args) {
Student Object1 = new Student();
Student Object2 = new Student();
Object1.change();
Student.id = 45;
Object2.change();
}
}
```

Output

101

45

Local variables

Are variables declared inside method body, constructor or block. Local variables are initialized when method, constructor or block start and will be destroyed once it end. Local variable reside in stack. Access modifiers are not used for local variable.

```
float getDiscount(int price){
float discount;
discount=price*(20/100);
return discount;
}
```

Here **discount** is a local variable.

Variable Scope in Java

Scope of a variable decides its accessibility throughout the program. As we have seen variables are different types so they have their own scope.

1. **Local variable**: Scope of local variable is limited to the block in which it is declared. For example, a variables declared inside a function will be accessible only within this function.
2. **Instance variable**: scope of instance variable depends on the access-modifiers **(public, private, default)**.
    i. If variable is declared as **private** then it is accessible **within class only**.
    ii. If variable is declared as **public** then it is **accessible for all and throughout the application.**
    iii. If variable is declared as **default** then it is **accessible within the same package.**
    iv. **Protected**: Protected has scope within the package and all sub classes

Variable Scope in Java

Java also supports many non-access modifiers, such as static, abstract, synchronized, native, volatile, transient etc. We will handle modifiers in details in our other classes ahead.

Datatypes in Java

Datatypes in Java

is a classification of data according to size and the type of values that can be stored in an identifier. Java has a rich implementation of data types

In java, data types are classified into two categories :-

1. Primitive Data type
2. Non-Primitive Data type

1. Primitive Data type

A **primitive data type** is either a data type that is built into a programming language, or one that could be characterized as a basic structure for building a program. A primitive data type can be of eight types :

char	boolean	byte	short	int	long	float	double

These eight primitive type can be put into four groups

1. Primitive Data type+

These eight primitive type can be put into four groups

1. Integer. This group includes byte, short, int, long

byte : It is 1 byte(8-bits) integer data type. Value range from -128 to 127. Default value zero. example: byte b=10;

short : It is 2 bytes(16-bits) integer data type. Value range from -32,768 to 32,767. Default value zero. example: short s=11;

int : It is 4 bytes(32-bits) integer data type. Value range from -2,147,483,648 to 2,147,483,647. Default value zero. example: int i=10;

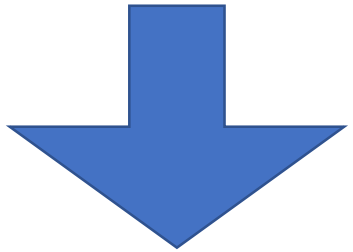
1. Primitive Data type++

These eight primitive type can be put into four groups

1. Integer. This group includes byte, short, int, long

long : It is 8 bytes(64-bits) integer data type. Value range from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807. Default value zero. example: long l=100012;

Example:



1. Primitive Data type+++

These eight primitive type can be put into four groups

1. Integer

Example Program

```
package corejava;
public class sample{
public static void main(String[] args) {
// byte type
byte b = 40;
System.out.println("b = "+b);
// short type
short k = 34;
System.out.println("k = "+k);
// int type
int m = 50;
System.out.println("m = "+m);
// long type
long l = 20;
System.out.println("l= "+l);
}
}
```

Output

b = 40
K = 34
m = 50
l = 20

1. Primitive Data type++++

These eight primitive type can be put into four groups

2. Floating point Numbers. This group is categorized into two,

includes float, double

i. float : It is 4 bytes(32-bits) float data type. Default value 0.0f. example: float ff=10.3f;

ii. double : It is 8 bytes(64-bits) float data type. Default value 0.0d. example: double db=11.123;

1. Primitive Data type++++

These eight primitive type can be put into four groups

2. Floating point Numbers

Example program;

```
public class sample{
public static void main(String[] args){
// float type
float f = 20.25f;
System.out.println("F = "+f);
 // double type
double d = 20.25;
System.out.println("D = "+d);
}
}
```

Output

F = 20.25

D = 20.25

1. Primitive Data type+++++

These eight primitive type can be put into four groups

3. Character This group represent char, which represent symbols in a character set, like letters and numbers.

i. char : It is 2 bytes(16-bits) unsigned unicode character. Range 0 to 65,535. example: char c='a';

Char Type Example

Char type in Java uses 2 bytes to unicode characters. Since it works with unicode then we can store alphabet character, currency character or other characters that are comes under the unicode set.

1. Primitive Data type+++++

These eight primitive type can be put into four groups

3. Character

Output

S
&
$

```
public class sample {
public static void main(String[] args){
char ch = 'S';
System.out.println(ch);
char ch2 = '&';
System.out.println(ch2);
char ch3 = '$';
System.out.println(ch3);
}
}
```

1. Primitive Data type++++++

These eight primitive type can be put into four groups

4. Boolean

This group represent boolean, which is a special type for representing true/false values.

They are defined constant of the language. example: boolean b=true;

Java works with two values only either true or false

1. Primitive Data type++++++

These eight primitive type can be put into four groups

4. Boolean program example

```
public class sample {
public static void main(String[] args){
boolean t = true;
System.out.println(t);
boolean f = false;
System.out.println(f);
 }
}
```

Output

true

false

2. Non-Primitive(Reference) Data type

A reference data type is used to refer to an object. A reference variable is declared to be of specific and that type can never be change.

For example.

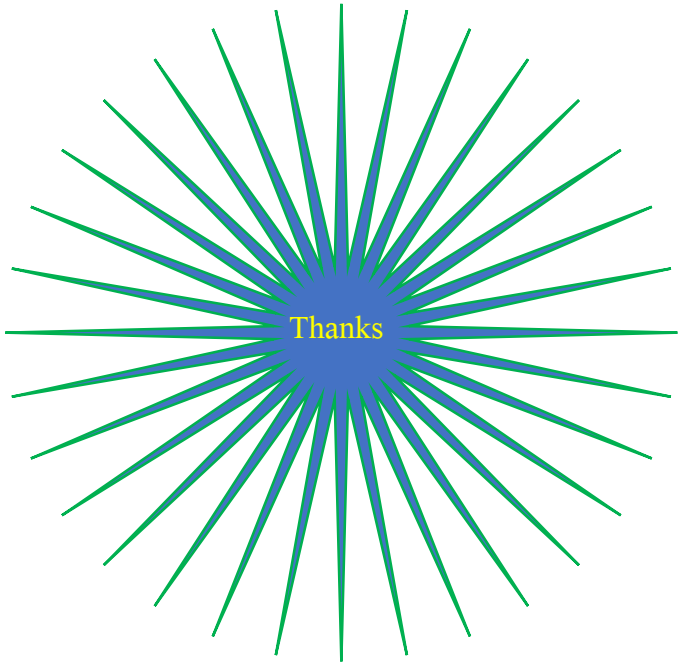
String str, here str is a reference variable of type String. String is a class in Java.

Reference type are used to hold reference of an object. Object can be instance of any class or entity. Details about reference type will be handled in classes and Object Lectures.

Summary

1. Identifiers,
2. Keywords
3. Writing java program(structure of java program)
4. Data types and
5. Variables in Java

Thank you for Listening



Reference

Techopedia. (2013, July 19). *What is a primitive data type? - definition from Techopedia*. Techopedia.com. Retrieved April 3, 2022, from https://www.techopedia.com/definition/29494/primitive-data-type

Variable in Java. Studytonight.com. (n.d.). Retrieved April 1, 2022, from https://www.studytonight.com/java/variable.php