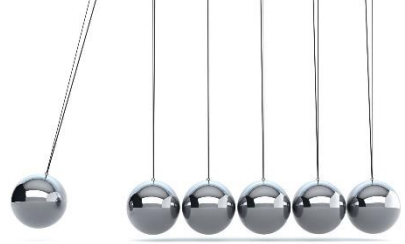
Java Fundamentals

24 JULY,2023

expleo

Outline



- Introduction to Programming paradigm
- Need – OOPs , OOPs Concept Realization
- Java Introduction
- Java Tokens
- Control Flow Statements
- Arrays
- Functions

expleo

Introduction to Programming Paradigm

Overview

- ✓ Way or style of programming.
- ✓ Defines the methodology of designing and implementing programs.
- ✓ **Three basic types**

1. **Unstructured Programming**: It contains only **one section, i.e., main**. The main consists of sequence of commands and statements i.e. unable to use **selection and repetition control statement** constructs.

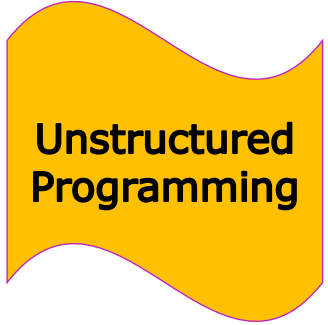
Overview

- ✓ **Three basic types**

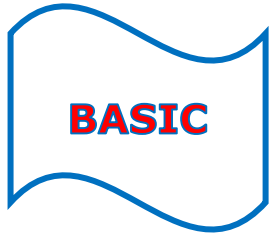
2. Structured Programming: It uses the structured control flow constructs of **selection** (if/then/else) and **repetition** (while and for), block structures, and **subroutines**.

3. Object Oriented Programming: Organized **around objects.** It combine **data** and **action** together.

Unstructured Programming



- **Unstructured programming** is the historically earliest programming paradigm capable of creating Turing-complete algorithms.
- It uses unstructured **jumps** to labels or instruction addresses with the use of unstructured control flow using **goto** statements or equivalent.
- Unstructured programming has been heavily criticized for producing **hardly-readable** ("spaghetti") code.



Unstructured Programming

Advantages

- Simple and easy to practice by novice programmer
- Good for small program; no lengthy planning needed.
- Speed and flexible
- Provides full freedom to programmers to program as they want.

Disadvantages

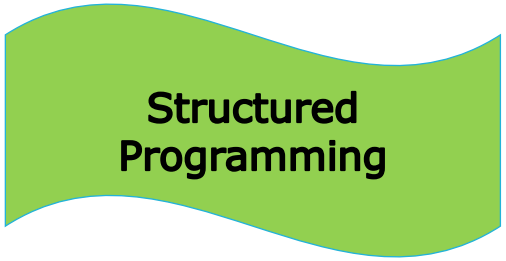
- No code reusability
- Cumbersome Maintenance
- Debugging is tough

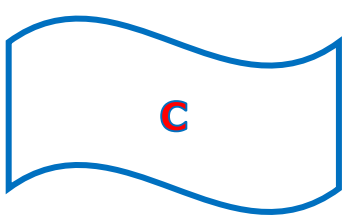
Structured Programming

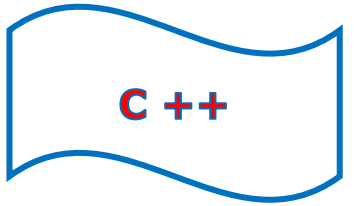
- Aimed at **improving the clarity, quality, and development time** of a computer program by making extensive use of the structured control flow constructs of selection (if / then / else) and repetition (while and for), block structures, and **subroutines**.
- Structured programming enforces a logical structure on the program being written to make it more **efficient** and **easier to understand** and modify.

Structured Programming

- The problem is **bifurcated into number of small problems** known as procedures (Also alternatively referred as functions or methods).







Structured Programming

Advantages

- Complexity can be reduced using modularity (divide and conquer)
- Logical structures ensure clear flow of control.
- Modules can be re-used many times; thus, it saves time and increase reliability.
- Easier to update/fix the program by replacing individual modules rather than larger amount of code.

Disadvantages

- Data is not secure
- Code reusability is less
- Can support the software development projects up to a certain level of complexity.
- If complexity of the project goes beyond a limit, it becomes difficult to manage.

Object Oriented Programming

Need

Programmers need a programming model that enables them to work with real-world entities / objects.

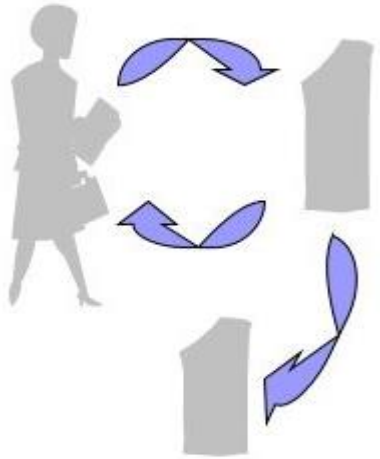
- People in real life have knowledge and can do a variety of tasks.
- Objects in OOP include fields to store **knowledge/state/data and methods** to do various tasks.

Structured Vs Object Oriented - Overview

Break down a programming task into a collection of **variables, data structures, and subroutines**.

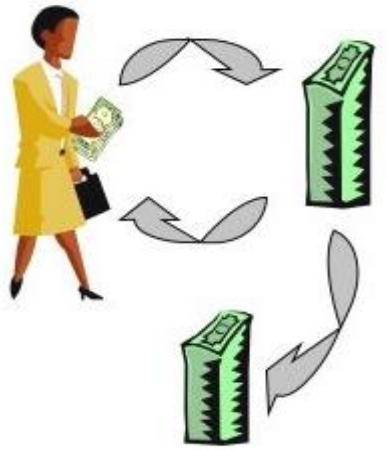
Break down a programming task into **objects** that expose behaviour (methods) and data (members or attributes).

Structured



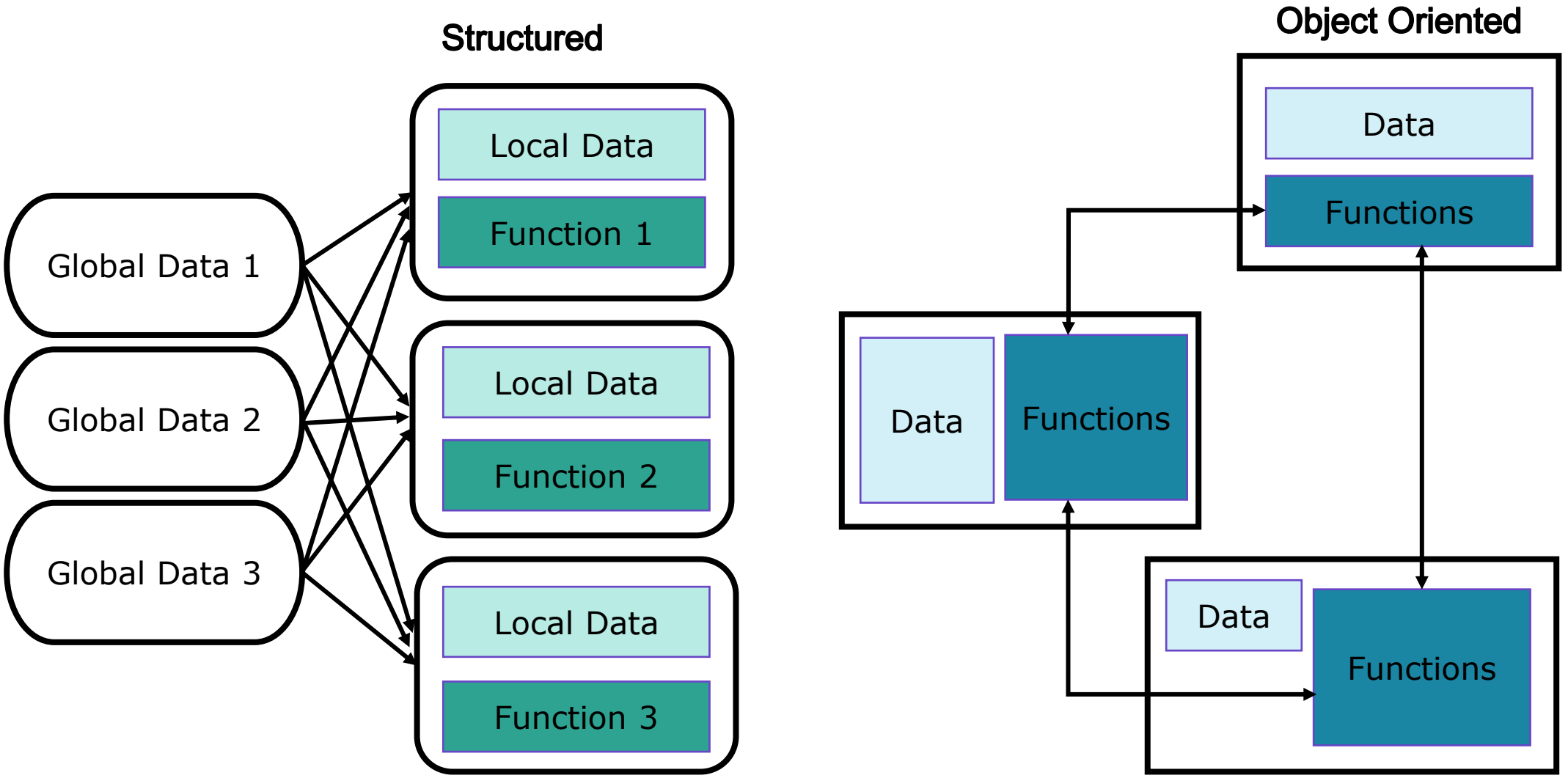
Subroutines:
Withdraw, Deposit, Transfer

Object Oriented

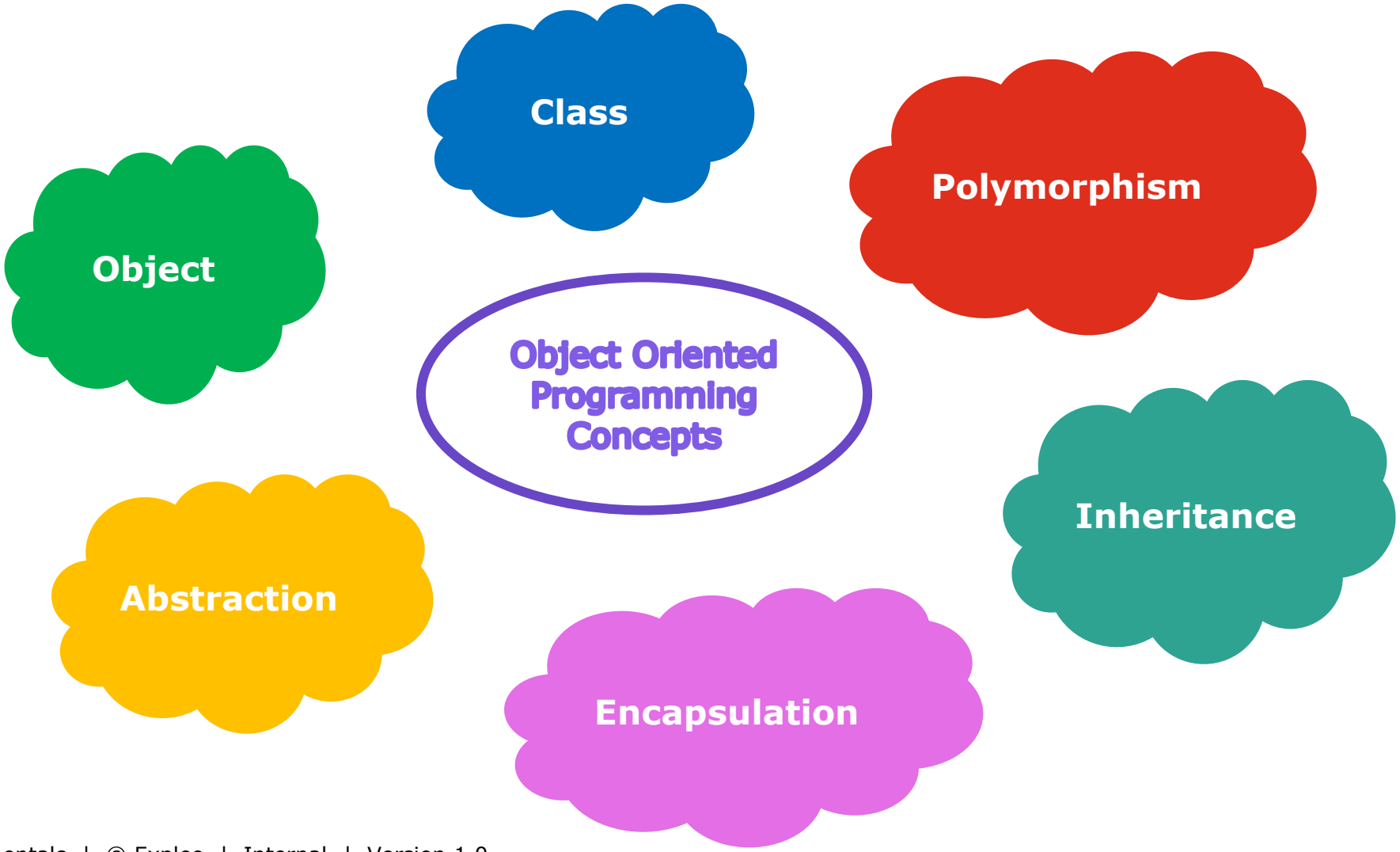


Objects:
Customer, Money, Account

Structured Vs Object Oriented - Overview



Features


Class
Polymorphism
Object
Object Oriented Programming Concepts
Inheritance
Abstraction
Encapsulation

Object - What it is?

- Object is the **key element** to understand *object-oriented* technology.
- It is a **real-world entity**.

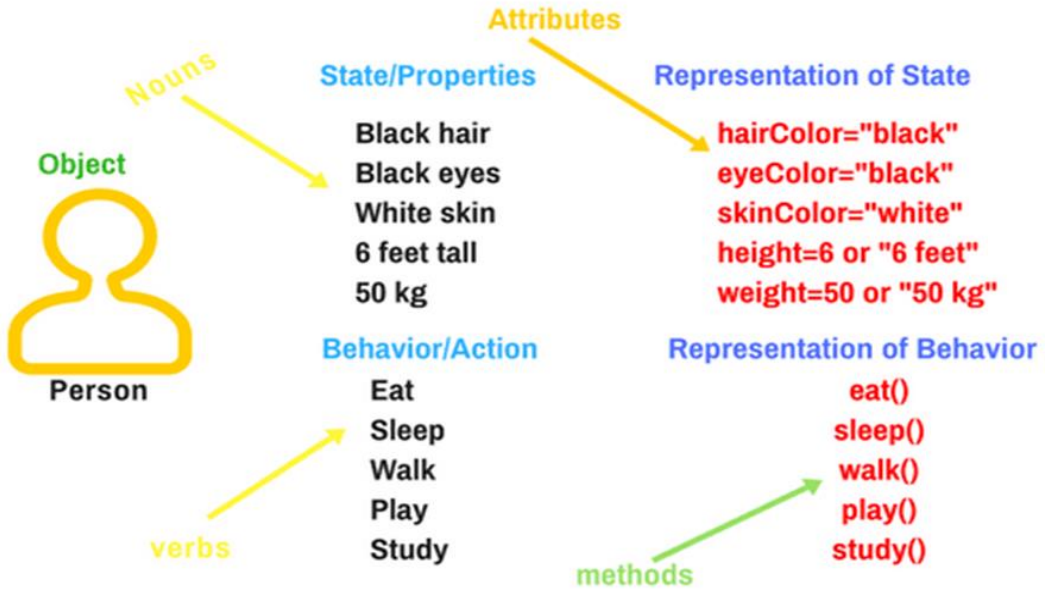
Real world Objects

Everything is an object in the real world

expleo

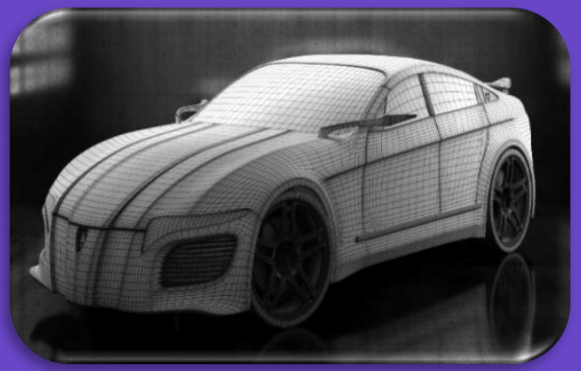
Object - What will it have?

- Real-world objects share **three characteristics**: State, Behavior and Identity.
  - The **state** of an object can be described as a set of attributes and their values.
  - The **behaviour /operation/action** of an object refers to the changes that occur in its attributes over a period.
  - **Identity** is the name that identifies the object.



Activity

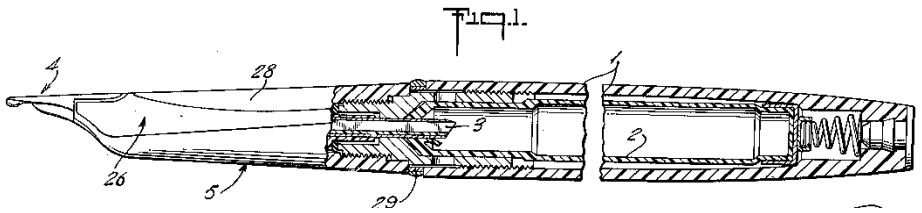
1. Identify the STATE, BEHAVIOUR and IDENTITY of the following Objects



expleo

Class

- Class is a **group of objects with similar properties**, common **behaviour**, common **relationships** with other objects, and common **semantics**.
- A class is the **template / blueprint** from which individual objects are created.



Object Oriented Programming

Object and Class

Class – Sportsman

States

- Name
- Sport
- Gender
- Height
- Weight
- rank

Behaviors

- Profiling
- BMI

Objects

Name: Cristiano Ronaldo
Sport: Football
Gender: Male
Height: 1.87m
Weight: 83kg
Rank: 6

Mr. Cristiano Ronaldo is a Football player with the international rank 6.

BMI is 23.7

Name: Rafael Nadal
Sport: Tennis
Gender: Male
Height: 1.85m
Weight: 85kg
Rank: 4

Mr. Rafael Nadal is a Tennis player with the international rank 4.

BMI is 24.6

Name: Mirabai Chanu
Sport: Weightlifting
Gender: Female
Height: 1.50m
Weight: 49kg
Rank: 2

Ms. Mirabai Chanu is a Weightlifting player with the international rank 2.

BMI is 21.8

expleo

Object – Instance of a class

Name: Cristiano Ronaldo
Sport: Football
Gender: Male
Height: 1.87m
Weight: 83kg
Rank: 6

Name: Rafael Nadal
Sport: Tennis
Gender: Male
Height: 1.85m
Weight: 85kg
Rank: 4

Name: Mirabai Chanu
Sport: Weightlifting
Gender: Female
Height: 1.50m
Weight: 49kg
Rank: 2

Instances of Sportsman class

Sports person 1

Sports person 2

Sports person 3

States

Name
Sport
Gender
Height
Weight
rank

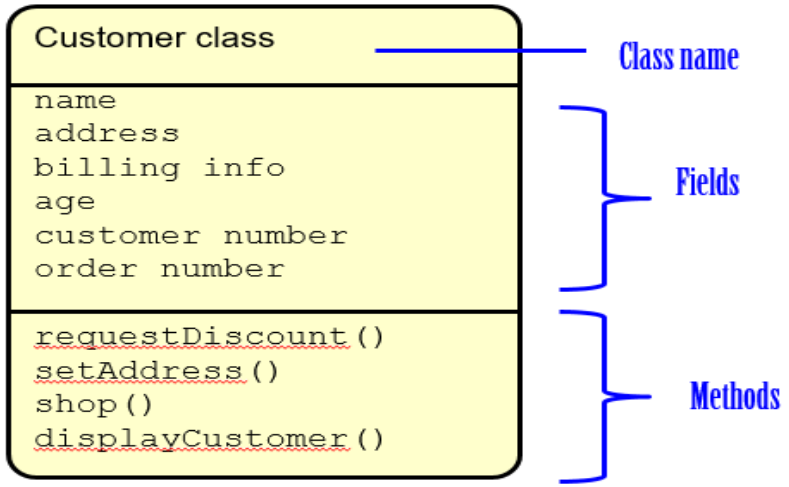
Behaviors

Profiling()
BMI()

Sportsman

Class diagram

Properties and Behaviors of 'Customer' class



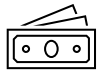
Properties:
- name
- address
- age
- order number
- customer number

Behaviors:
- shop()
- setAddress()
- requestDiscont()
- displayCustomer()

Abstraction

- Data abstraction is **providing** only what is **relevant/essential** information about the data to the outside world, **hiding** the **irrelevant/background details/implementation**.
  - **For Example,** when you **send an email** to someone you just click send and you get the success message.
  - **What actually happens when you click send?**
  - How data is transmitted over network to the recipient?
  - All these are hidden from you (**because it is irrelevant to you**).

Abstraction



ATM user need not to know what is the process behind the screen.

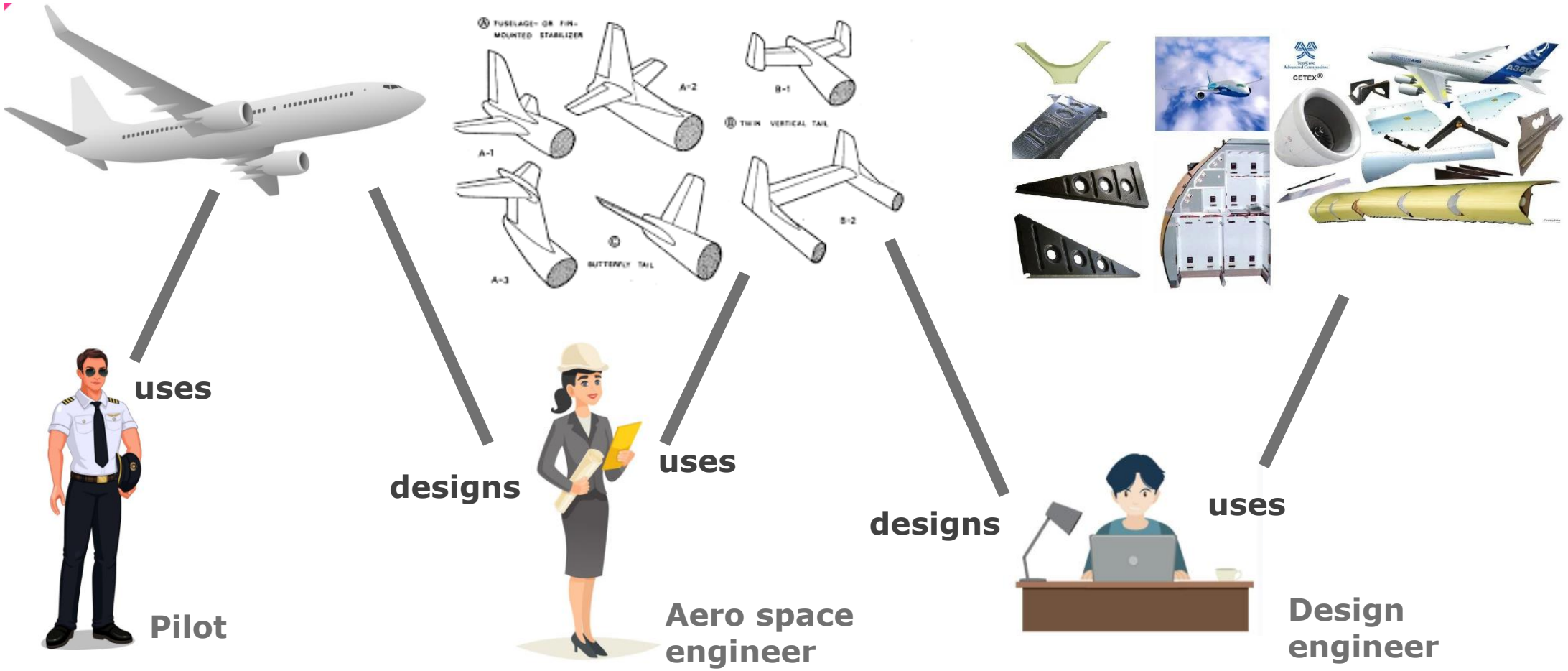
Musician need not to know how the sound is produced. All they need to know is which key to press

Note: An abstraction includes the essential details relative to the perspective of the user.

Abstraction

Note: An abstraction includes the essential details relative to the perspective of the user.

Abstraction

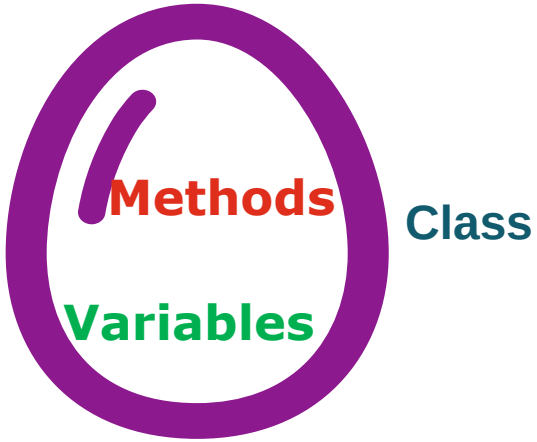


Note: An abstraction includes the essential details relative to the perspective of the user.

Encapsulation

- It refers to the **bundling of data with the methods** that operate on that data, or the restricting of direct access to some of an object's components.
- The internal representation of an object is generally **hidden from view** outside of the object's definition.
- It can reduce system complexity, and thus increase the robustness.

Medicines are not directly accessible / protected from outside



Encapsulation Vs Abstraction

Encapsulation

Abstraction

Note: Abstraction says **what details** to be made visible where as Encapsulation provides the **level of access** right to that visible details.

Inheritance

- Inheritance is a process in which **one object gets** the capability to acquires all the **properties and behaviours** of its parent object automatically.
- As a result, there is no need to rewrite the member again and again. This **'Code reusability'** feature avoids the **code redundancy.**

Inheritance

Grand Dad

House - owns

Inherited from

Dad

House - derives

Car - owns

Inherited from

Son

House - derives

Car - derives

Bike - owns

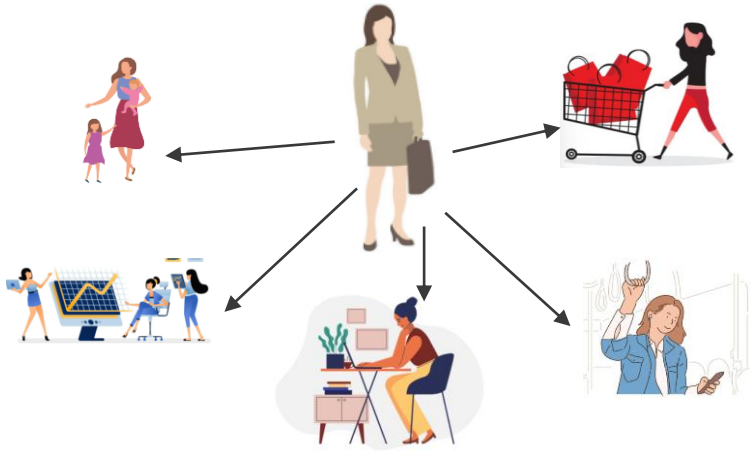
Polymorphism

- Polymorphism is the concept where an **object behaves differently in different situations.**
- More precisely we say it as **'many forms of single entity'.**

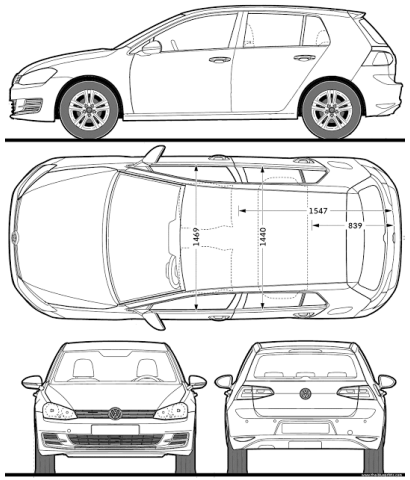
Polymorphism

Example 1: We turn on the computer by one button at the same time we can turn off the computer by the same button.

Example 2: A person might have a **variety of characteristics**. He is a man, and he can also be a father, a husband, or an employee. As a result, the same person behaves differently in different situations.



Summary: Car Object



CLASS

OBJECT

ENCAPSULATION

INHERITANCE

ABSTRACTION

POLYMORPHISM

Procedure Oriented Vs Object Oriented

POP	OOP
The program is divided into functions.	The program is divided into objects.
Top-down approach	Bottom-Up approach
Every function has different data, so there's no control over it.	Data in each object is controlled on its own.
Parts of a program are linked through parameter passing.	Object functions are linked through message passing.
Expanding data and function is not easy.	Adding new data and functions is easy.
Inheritance is not supported.	Inheritance is supported in public, private& protected modes.

Procedure Oriented Vs Object Oriented

POP	OOP
No access modifiers supported.	Access control is done with access modifiers.
No data hiding. Data is accessible globally.	Data can be hidden using Encapsulation.
Overloading is not possible.	Overloading can be done.
No friend function.	No friend function except C++.
No virtual classes or functions.	The virtual function appears during inheritance.
No code reusability.	The existing code can be reused.
Not suitable for solving big problems.	Used for solving big problems.

Quiz

1. Identify equivalent OOP Principle for the following scenarios.

A person who understands more than two languages yet can converse in any of them depending on the situation.

Polymorphism

Quiz

2. Identify equivalent OOP Principle for the following scenarios.

A scientific calculator is a more advanced version of a regular calculator.

Inheritance

Quiz

3. Identify equivalent OOP Principle for the following scenarios.

When you go to the bank and tell the banker(Object) to make a deposit(Method) of X dollars into your account(object), the banker will ask for your account number as well as the amount of cash you want to deposit. After a few moments, the banker informs you. Hey, your deposit was completed successfully, and your receipt is attached.

Abstraction

Quiz

4. How many objects can you identify in the below picture?

a) 16

b) 2

c) 4

d) 8

Ans: a) 16

Quiz

5. Is ‘Sound’ an object?

a) Yes

b) No

Ans: a) Yes

Quiz

6. Class is a collection of object

a) True

b) False

Ans: b) False

Quiz

7. Which of the following statement is correct?

a) Class is an instance of object.

b) Object is an instance of a class.

c) Class is an instance of data type.

d) Object is an instance of data type.

Ans: b)

Quiz

8. Which of the following concepts means wrapping up of data and functions together?

a) Abstraction

b) Encapsulation

c) Inheritance

d) Polymorphism

Ans: b)

Introduction to Java

Java

- Java is a **general purpose, object-oriented, concurrent, and secured** programming language.
- It is a **widely used robust technology**.
- It is a **platform** – It has a runtime environment **(JRE)** and **API.**
- **James Gosling –** the father of Java.

History

- Java initial work started in **1990 by Sun Microsystems** engineer **Patrick Naughton** as a part of the **Stealth Project.**
- The Stealth Project soon changed to the **Green Project**, with **Mike Sheridan and James Gosling** joining the ranks, and the group began developing new technology for programming next-generation smart appliances.
- **James Gosling**, **Mike Sheridan**, and **Patrick Naughton** initiated the Java language project in **June 1991.**
- Gosling attempted to modify and **extend C++**, but quickly abandoned this approach in favour of creating an entirely new language.

History

- Firstly, it was called "**Greentalk**" by James Gosling, and file extension was .gt. After that, it was called **Oak**, named after the tree that stood outside his office.
- Originally **designed for small, embedded systems in electronic appliances** like set-top boxes. Then incorporate some changes based on emergence of **World Wide Web**, which demanded **portable programs**.
- In 1995, Oak was renamed as **"Java"** because it was already a trademark by Oak Technologies.
- The first publicly available version of **Java (Java 1.0)** was released in 1995.
- In 2006 Sun started to make Java available under the **GNU General Public License** (GPL). Sun Microsystems was acquired by the **Oracle Corporation in 2010**. Oracle continues this project called OpenJDK.

Where it is used?

- Java technology is used by more than 6 million developers and runs on more than 5.5 billion devices.
- It is used create desktop Applications, Web Applications, Enterprise Applications such as banking applications, mobile applications.
- Web services and Cloud based applications.
- Embedded System
- Games etc.,

Editions

J2SE	J2ME	J2EE
• **Java 2 Standard Edition** • Client side standalone applications	• **Java 2 Micro Edition** • Applications for mobile Devices	• **Jakarta EE/Java 2 Enterprise Edition** • Applications for Enterprise

Version

Version	class file format version[7]	Release date	End of Free Public Updates[8][9][10][11][12][13][14]	Extended Support Until
JDK 1.0	45	23rd January 1996	May 1996	—
JDK 1.1	45.3	2nd February 1997	October 2002	?
J2SE 1.2	46	4th December 1998	September 2003	?
J2SE 1.3	47	8th May 2000	October 2010	?
J2SE 1.4	48	13th February 2002	October 2008	February 2013
Java SE 5	49	29th September 2004	November 2009	April 2015
Java SE 6	50	11th December 2006	April 2013	December 2018 for Oracle[8] December 2026 for Azul[11]
Java SE 7	51	28th July 2011	September 2022 for OpenJDK Maintained by Oracle until May 2015[15], Red Hat until August 2020[16] and Azul until September 2022[17]	July 2022 for Oracle[8] June 2020 for Red Hat[12] December 2027 for Azul[11]
Java SE 8 (LTS)	52	18th March 2014	(OpenJDK currently maintained by Red Hat)[18] March 2022 for Oracle (commercial) December 2030 for Oracle (non-commercial) December 2030 for Azul[11] May 2026 for IBM Semeru[13] At least May 2026 for Eclipse Adoptium[9] At least May 2026 for Amazon Corretto[10]	December 2030 for Oracle[8] November 2026 for Red Hat[12]
Java SE 9	53	21st September 2017	March 2018 for OpenJDK	—
Java SE 10	54	20th March 2018	September 2018 for OpenJDK	—
Java SE 11 (LTS)	55	25th September 2018	(OpenJDK currently maintained by Red Hat)[19] September 2026 for Azul[11] October 2024 for IBM Semeru[13] At least October 2024 for Eclipse Adoptium[9] At least September 2027 for Amazon Corretto[10] At least October 2024 for Microsoft[20][14]	September 2026 for Oracle[8] September 2026 for Azul[11] October 2024 for Red Hat[12]
Java SE 12	56	19th March 2019	September 2019 for OpenJDK	—
Java SE 13	57	17th September 2019	(OpenJDK currently maintained by Azul)[21] March 2023 for Azul[11]	—
Java SE 14	58	17th March 2020	September 2020 for OpenJDK	—
Java SE 15	59	16th September 2020	(OpenJDK currently maintained by Azul)[22] March 2023 for Azul[11]	—
Java SE 16	60	16th March 2021	September 2021 for OpenJDK	—
Java SE 17 (LTS)	61	14th September 2021	(OpenJDK currently maintained by SAP)[23] September 2029 for Azul[11] October 2027 for IBM Semeru[13] At least September 2027 for Microsoft[14] At least September 2027 for Eclipse Adoptium [9]	September 2029 or later for Oracle[8] September 2029 for Azul[11] October 2027 for Red Hat[12]
Java SE 18	62	22nd March 2022	September 2022 for OpenJDK and Adoptium	—
Java SE 19	63	20th September 2022	March 2023 for OpenJDK	—
Java SE 20	64	21st March 2023	September 2023 for OpenJDK	—
Java SE 21 (LTS)	65	September 2023	September 2028	September 2031 for Oracle[8]

Legend: Old version | Older version, still maintained | Latest version | Future release

Environment

- Java includes many development tools, classes and methods.
- **Development tools** are part of Java Development Kit **(JDK)**
- The classes and methods are part of Java Standard Library **(JSL),** also known as Application Programming Interface **(API).**
- It includes **hundreds of classes** and **methods grouped** into **several packages** according to their functionality.

Environment

- **Java Development Kit (JDK) –** Development environment to build java applications

JDK = Java Runtime Environment (JRE) + Development Tool

- **Java Runtime Environment (JRE) –** It provides the minimum requirements for executing a java application

JRE = Java Virtual Machine (JVM) + Library Classes

Environment

- **Java Virtual Machine (JVM)** – **Code execution component** of java application. It **provides runtime environment** in which **java byte code** can be executed.

Java Interpreter + Just-In-Time Compiler

Note

- JVM, JRE and JDK are **ported to different platforms** to provide **hardware** and **operating system-independence.**

Environment

Features

- **Simple –** Java syntax based on earlier languages like C and C++. Designed considering the **pitfalls** of earlier languages
- **Object Oriented** – Java supports all the **Object Oriented features**.
- **Secured – No explicit pointer** support, run inside the java **virtual machine sandbox.** The **class loader, byte code verifier and security manager** component of JVM enable the secured environment. Java also supports application developer to use other security mechanism like **SSL, JAAS, Cryptography**, etc.

Features

- **Robust –** Java have strong memory management, lack of pointers that avoids security problems, automatic garbage collection, exception handling and the type checking.
- **Platform Independent** - Once compiled ,code will be run on any platform without recompiling or any kind of modification **-"Write Once Run Anywhere(WORA)".** This is made possible by making use of a **Java Virtual Machine(JVM).**

Note

- JVM, JRE and JDK are **ported to different platforms** to provide **hardware** and **operating system-independence.**

Features

- **Architecture-neutral –** the size of **primitive types is fixed**
- **Distributed** – Java enable to create distributed application with the help of RMI and EJB
- **Multi-threaded –** Java run application concurrently.

Environmental Setup

- To **run java applications**, we need to prepare the **environmental setup**. For installing Java we need the following:
- The Java Runtime Environment **(JRE)** (Includes Java Virtual Machine (JVM))
- The Java Developer Kit **(JDK)** (Includes the Java Compiler)
- A text editor
- If JDK is installed, you automatically get the JRE
    - It can be **downloaded** here:

      https://www.oracle.com/in/java/technologies/javase/jdk11-archive-downloads.html

Java Installation Steps

1. Go to the Java Downloads Page and click on the option of **Download.**

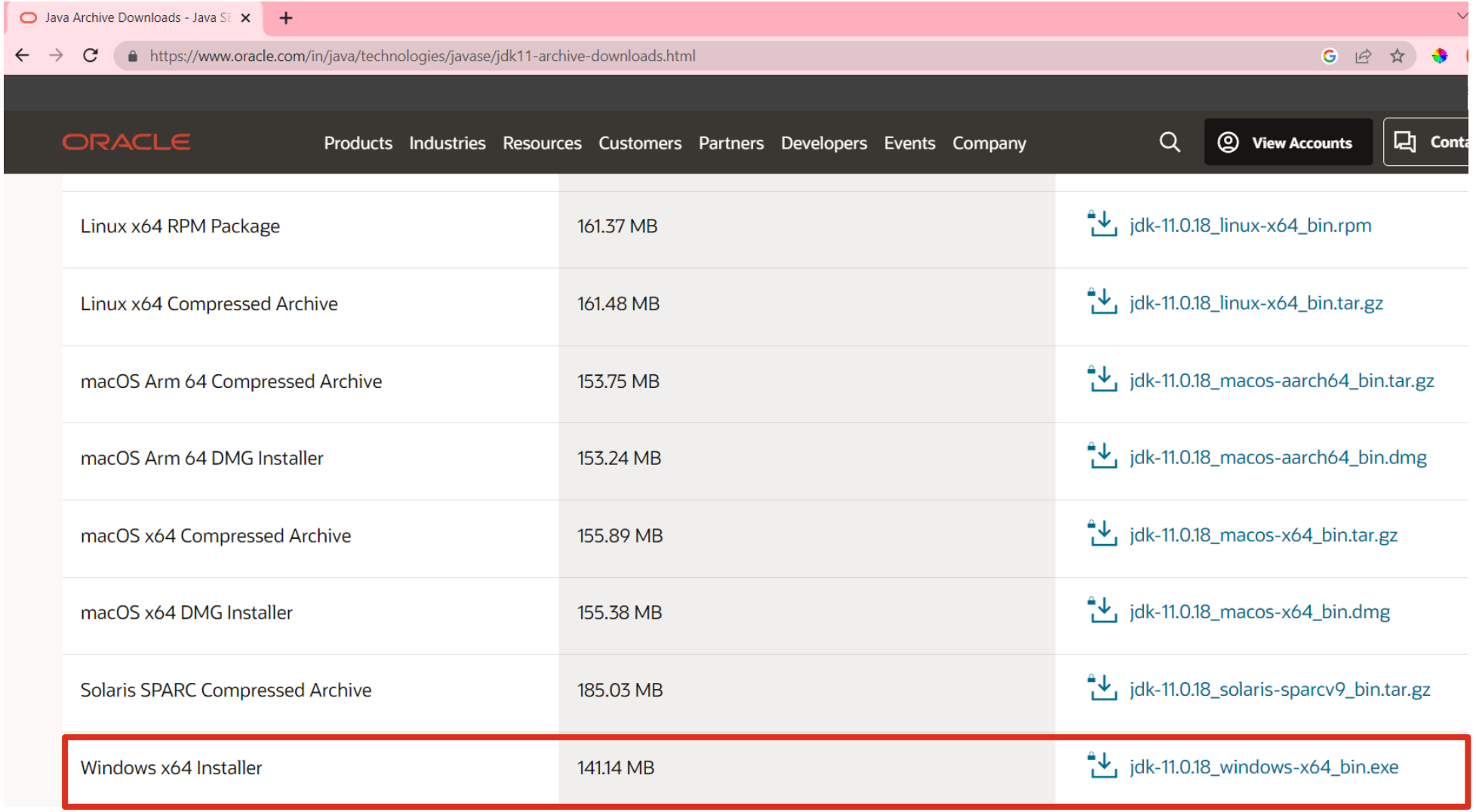
2. After clicking **Download**, you will be redirected to a page, select the **Accept License Agreement** button.

   Choose a version based on the operating system

   Here, I have chosen **jdk-11.0.18-windows-x64.exe**

3. Now, once the file is downloaded, run the installer and keep clicking on **Next**, till you get a message of finished downloading.

Java Installation Steps

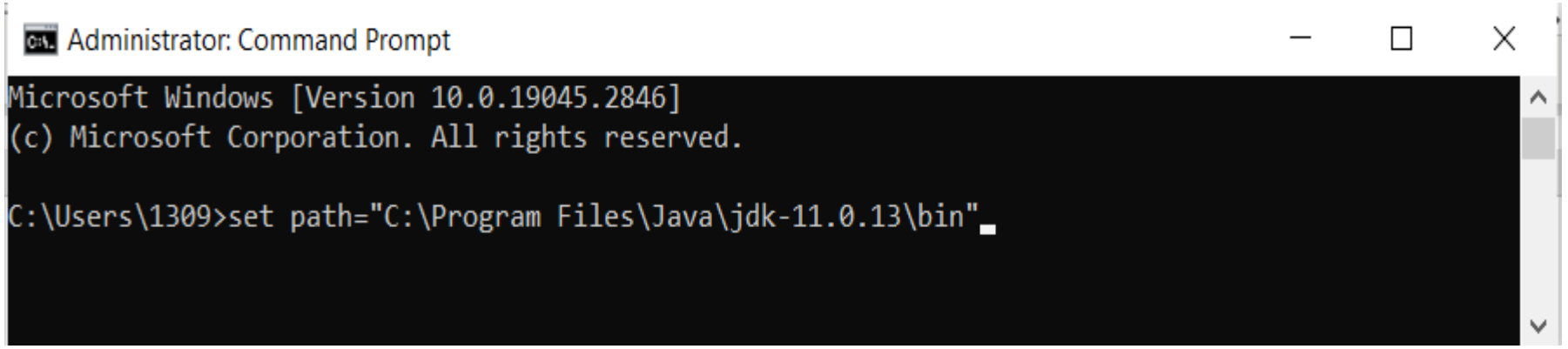


First Java Program in Command Prompt

- You can write the code using any one of the text editor like **notepad** and **execute in Command prompt,**
- Before executing the code, need to set the path for the Java file.
- The path is required to be set for using tools such as javac, java, etc.
- If we are saving the **Java source file inside the JDK/bin directory**, the path is not required to be set because all the tools will be available in the current directory.

First Java Program in Command Prompt

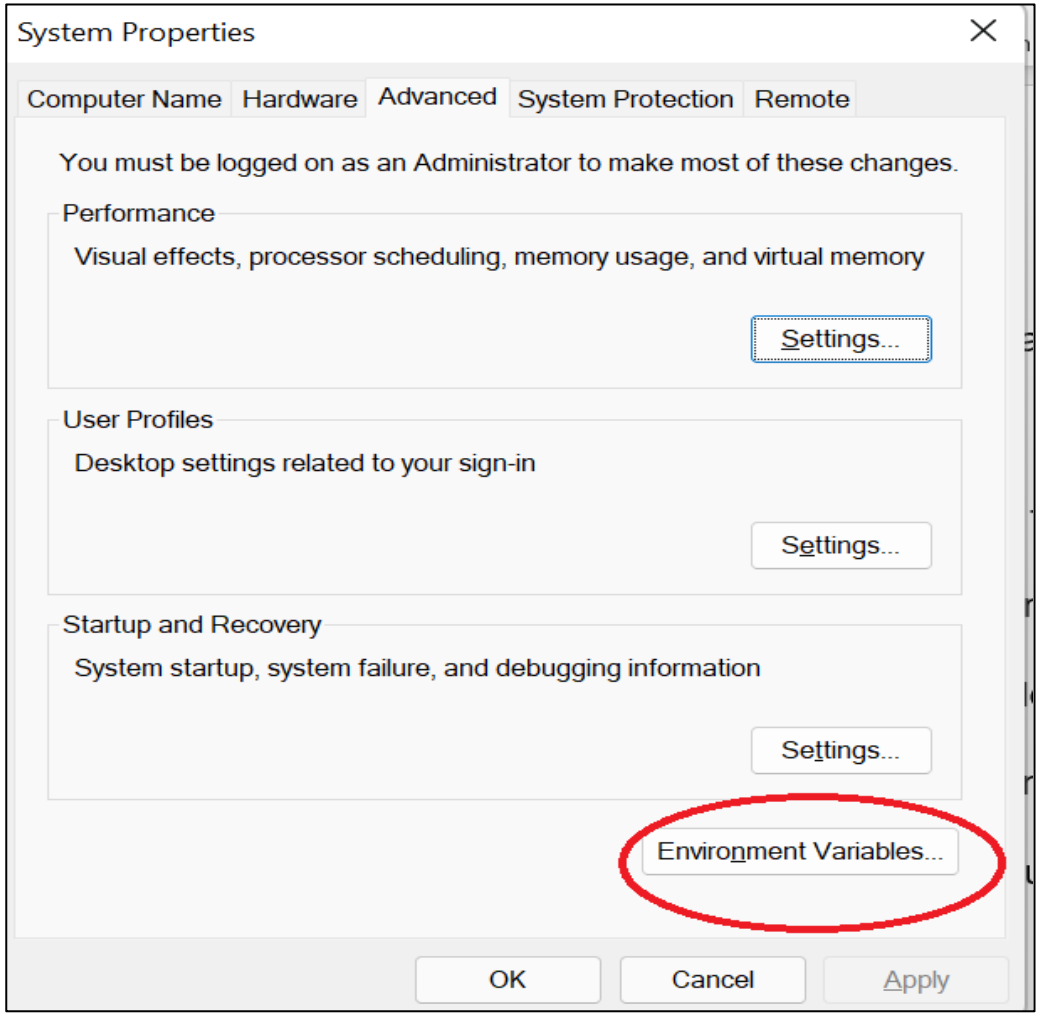
- If we have a **Java file outside the JDK/bin folder**, it is necessary to set the path of JDK.
- We can set the path of Java file in two ways as follows
  - **Temporary path**: Open CMD, copy the path of the JDK/Bin directory and write in the command prompt: set path = copied path(path where your software has installed).



First Java Program in Command Prompt

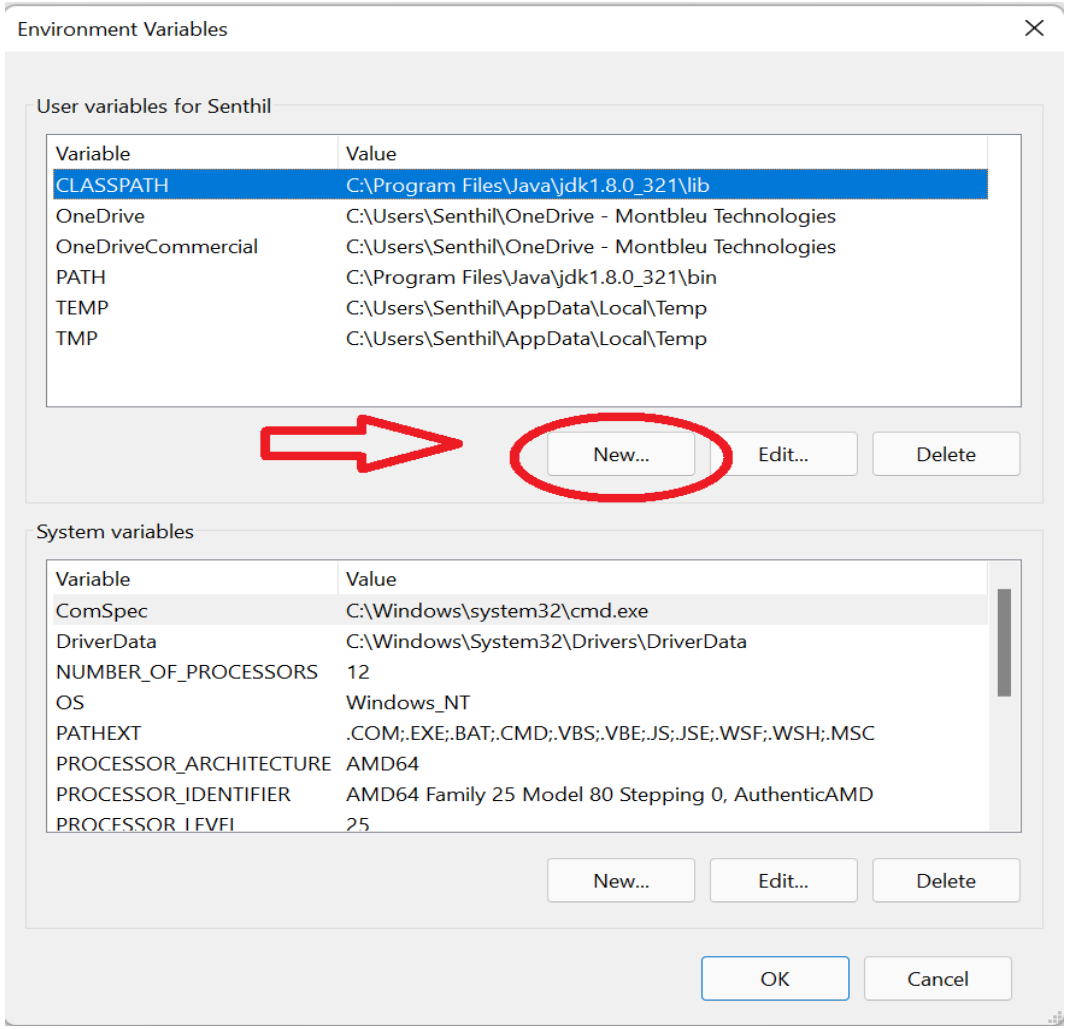
Permanent Path:

- Go to Start and search for '**System**'. Then, click on 'System' and go to Advanced System Settings.
- Now, click on '**Environment Variables**' under the 'Advanced' tab as shown here



Fist Java Program in Command Prompt

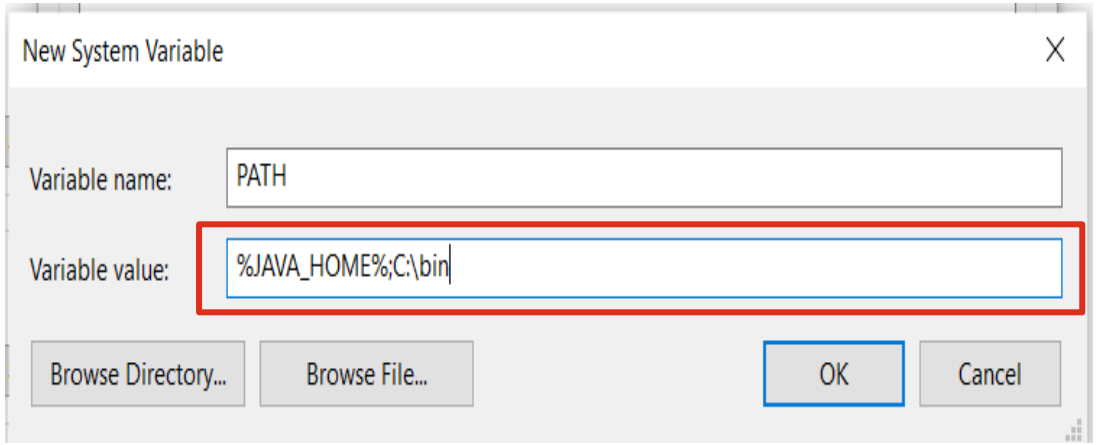
- Next, under **User Variables** choose New.
- Enter the variable name as **'JAVA_HOME'** and the full path to Java installation



expleo

First Java Program in Command Prompt

- Next, **edit the path** of the system variable as shown here.
- In the row of ‘Variable value’, enter the path of the folder(enter the path of the software have installed).
- Click OK .



First Java Program in Command Prompt

- You can either use temporary or permanent methods to set the path.
- Once have you completed, check the software have installed and the path is properly set by typing the command in CMD ***"java -version".***
- This command will display the **installed version of Java** in your system.

First Java Program in Command Prompt

Note:

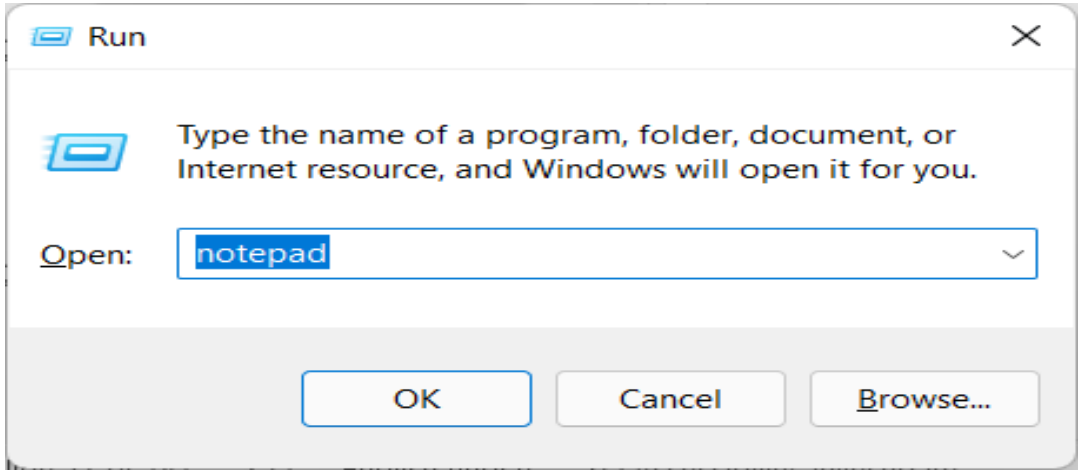
- If you set the path with the **permanent method**, then **no need to set the path for every individual java program**.
- But In the case of the **temporary method**, whenever you are opening the CMD you need to set the path**.**
- Once you have completed the path set continue with the execution of your code.

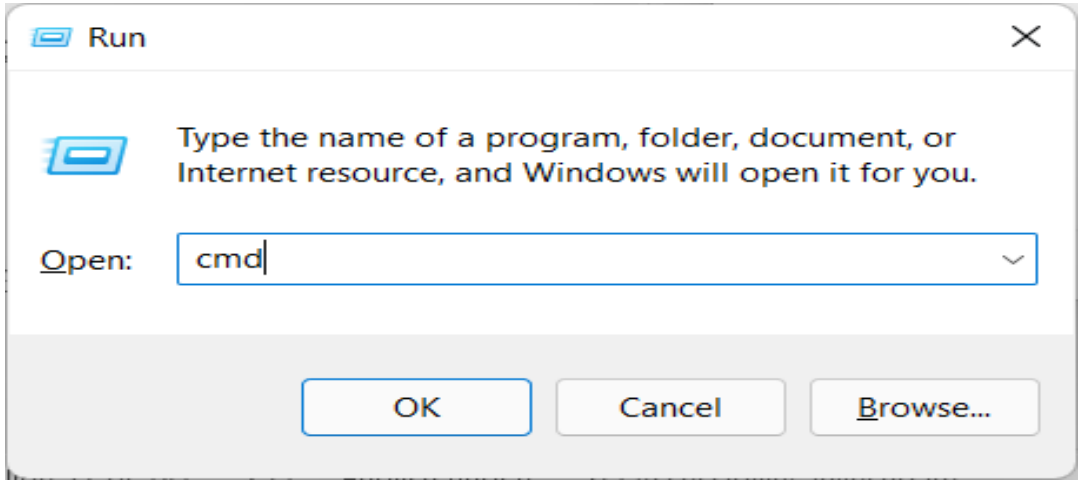
First Java Program in Command Prompt

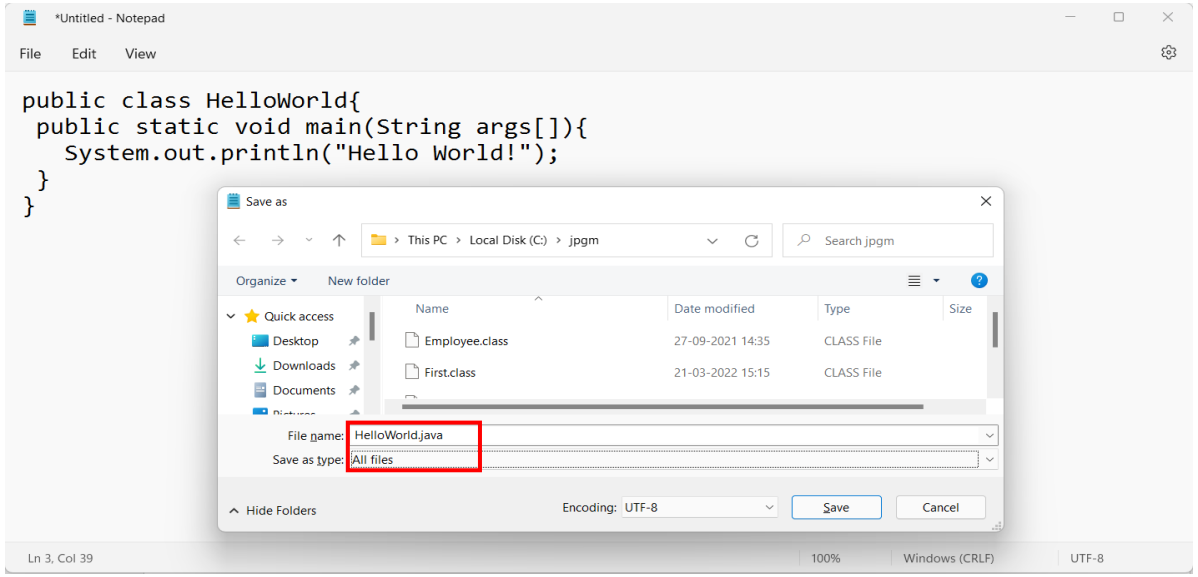
- To execute the Java program, the following steps need to be followed
  - **Step 1:** Open the notepad or press the *Windows* + *R*, type notepad, and click Ok.
  - **Step 2:** Write a Java program that you want to compile and run in the notepad.
  - **Step 3:** To save a Java program press Ctrl + S key and provide the file name. Remember that the file name must be the same as the **class name** followed by the **.java** extension.
    - **Example:** Writing the java program in the notepad with the class name *"HelloWorld"*

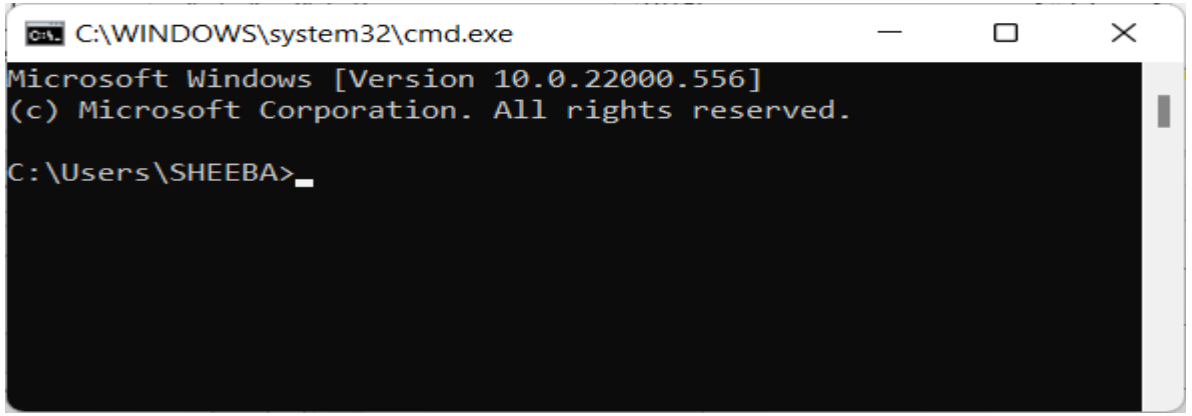
      in notepad and saving the file as **HelloWorld.Java.**
  - **Step 4:**To compile and run the program, open the **Command Prompt** by pressing **Windows Key + R**, type **CMD**, and press **enter** key or click on the **Ok** button.

First Java Program in Command Prompt









expleo

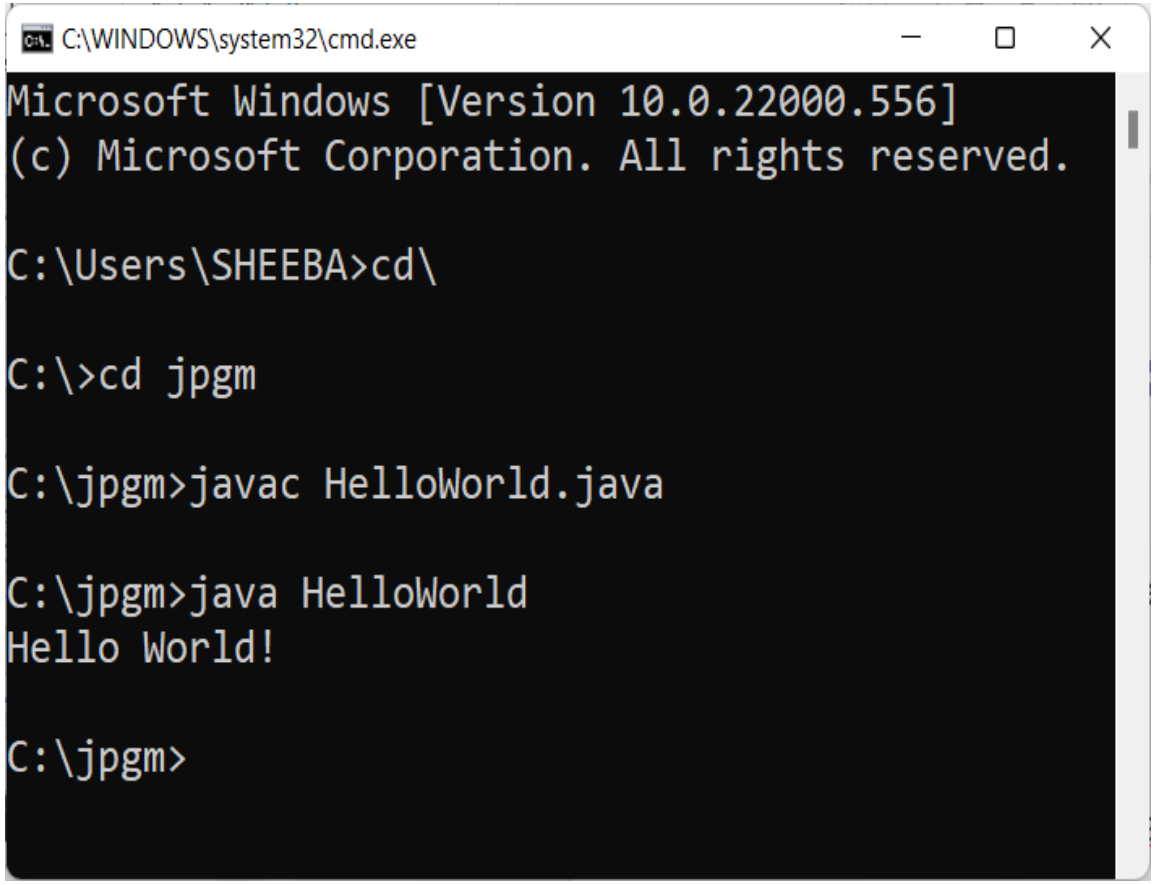
First Java Program in Command Prompt

- **Step 5:** To compile the Java program type the following comment

  - **Example:** **javac** HelloWorld.Java

- **Step 6:** To execute the Java program type the following comment

  - **Example:** **java** HelloWorld

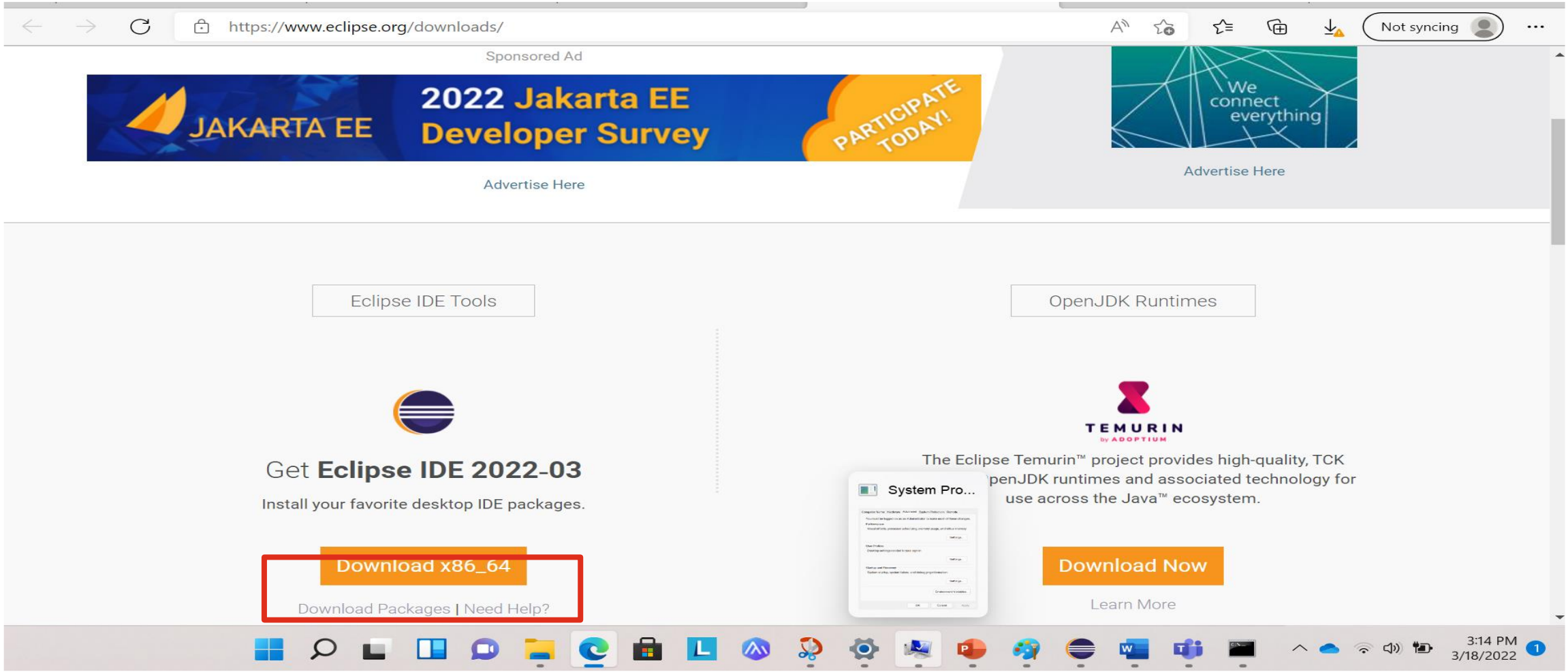


Eclipse IDE

- Using a text editor is the basic way of creating a Java program, but there are tools available to make it easier to create Java programs called **IDEs** like **Eclipse**.
- **Eclipse** is a **third-party open-source IDE** that is very powerful and popular.
    - It can be downloaded here: http://www.eclipse.org/downloads/

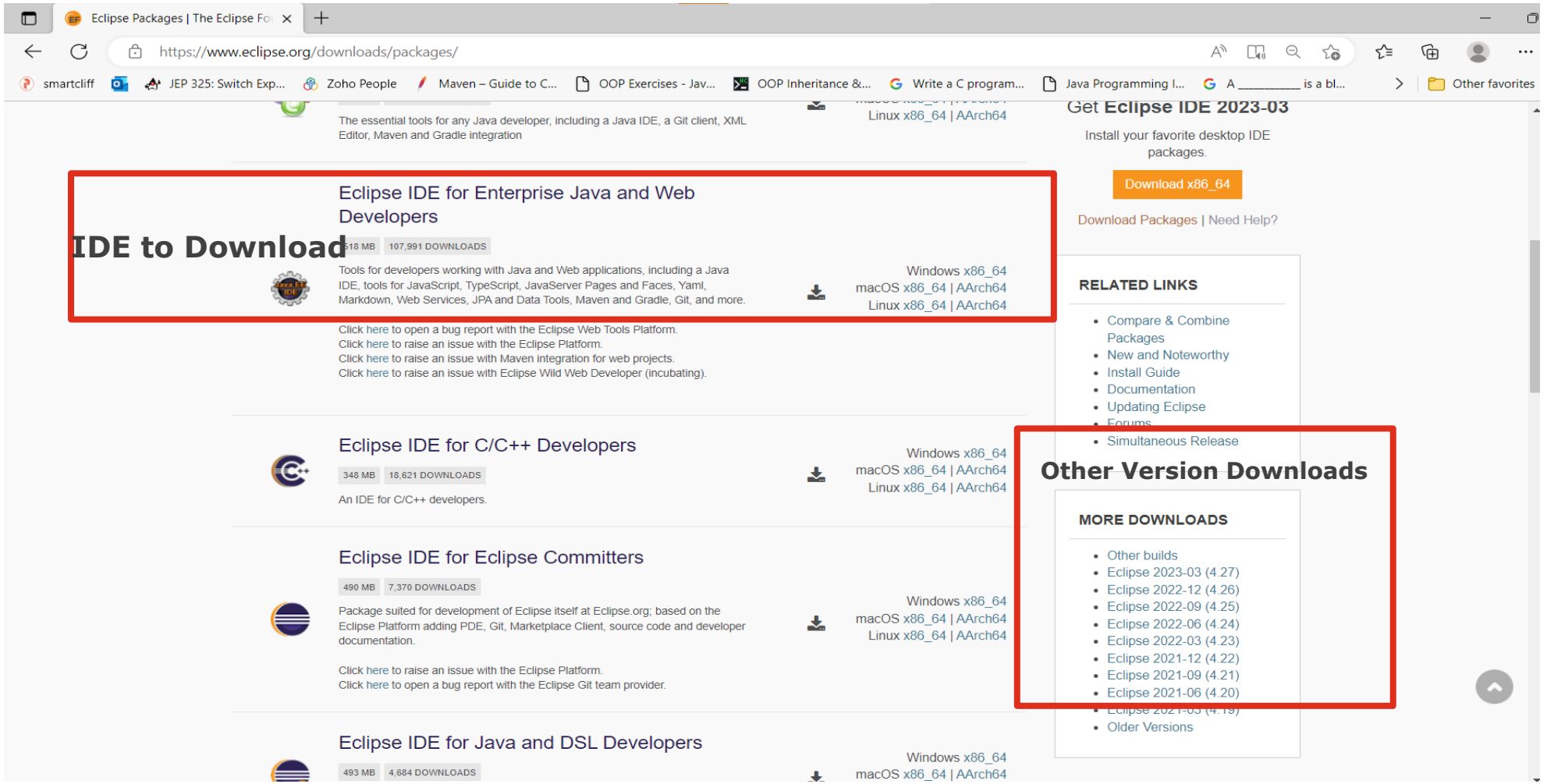
Eclipse IDE

Eclipse download page.



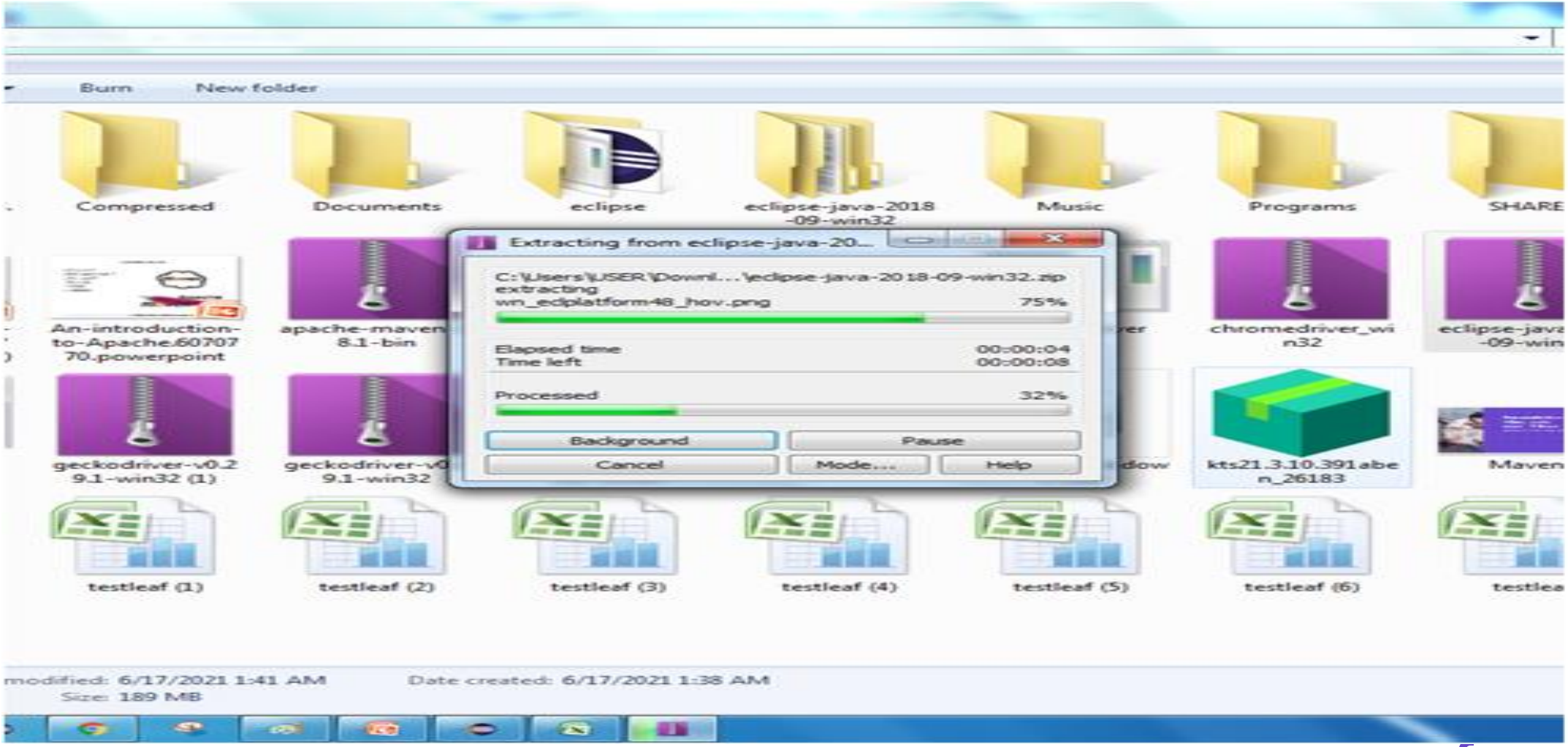
Eclipse IDE

Choose an option from the list



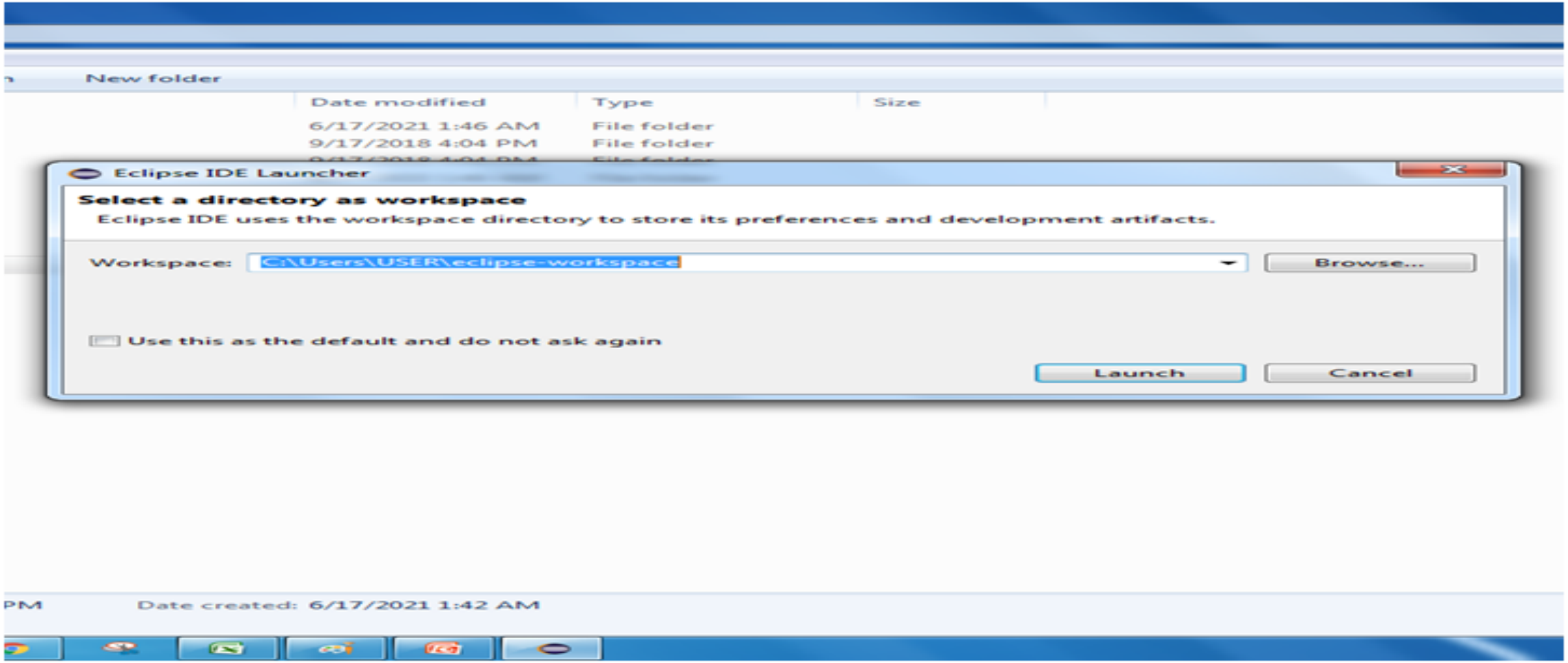
Eclipse IDE

After download, right click and extract the eclipse file



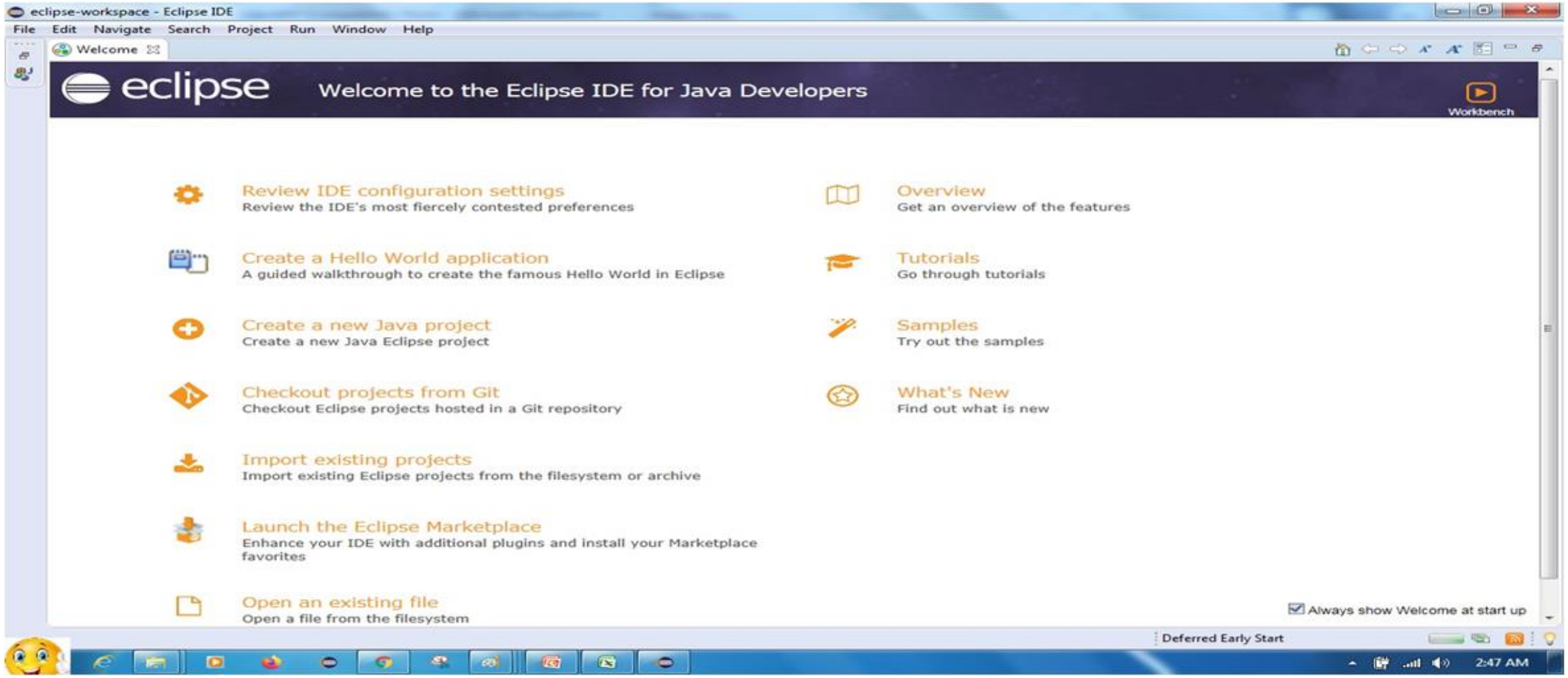
Eclipse IDE

Configuring Eclipse: Select a directory to create a workspace for projects



Eclipse IDE

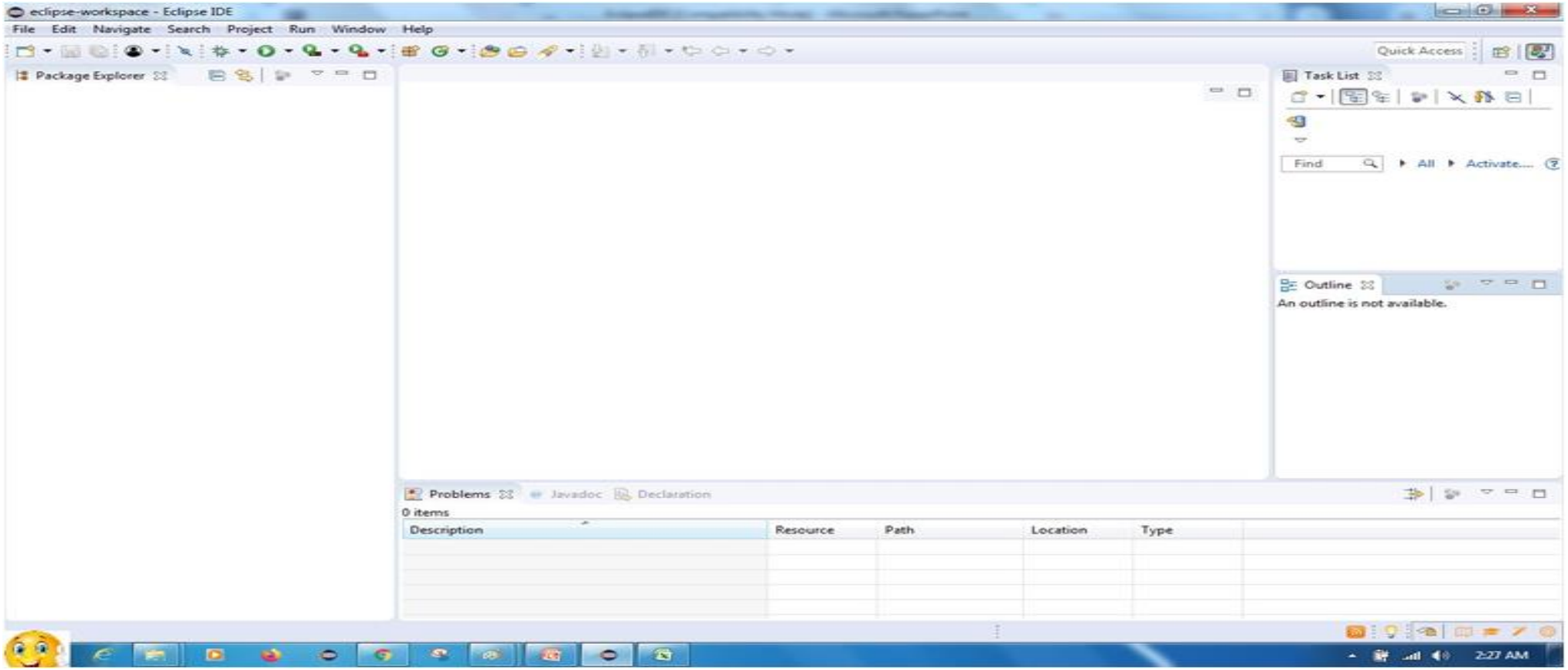
Once the workspace is chosen, you will get the welcome page



Eclipse IDE

After closing the welcome screen ,the project explorer and other options will be shown up.

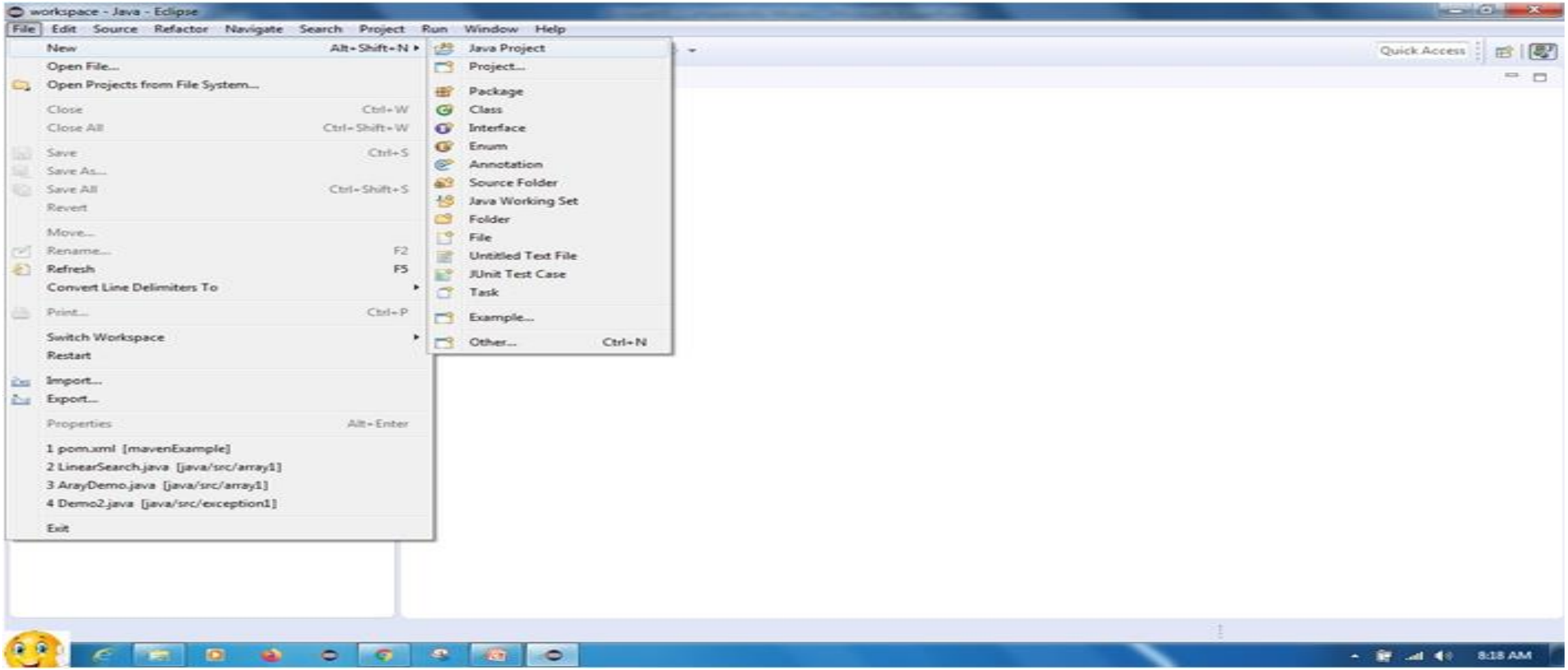
Now you ready are to start working.



expleo

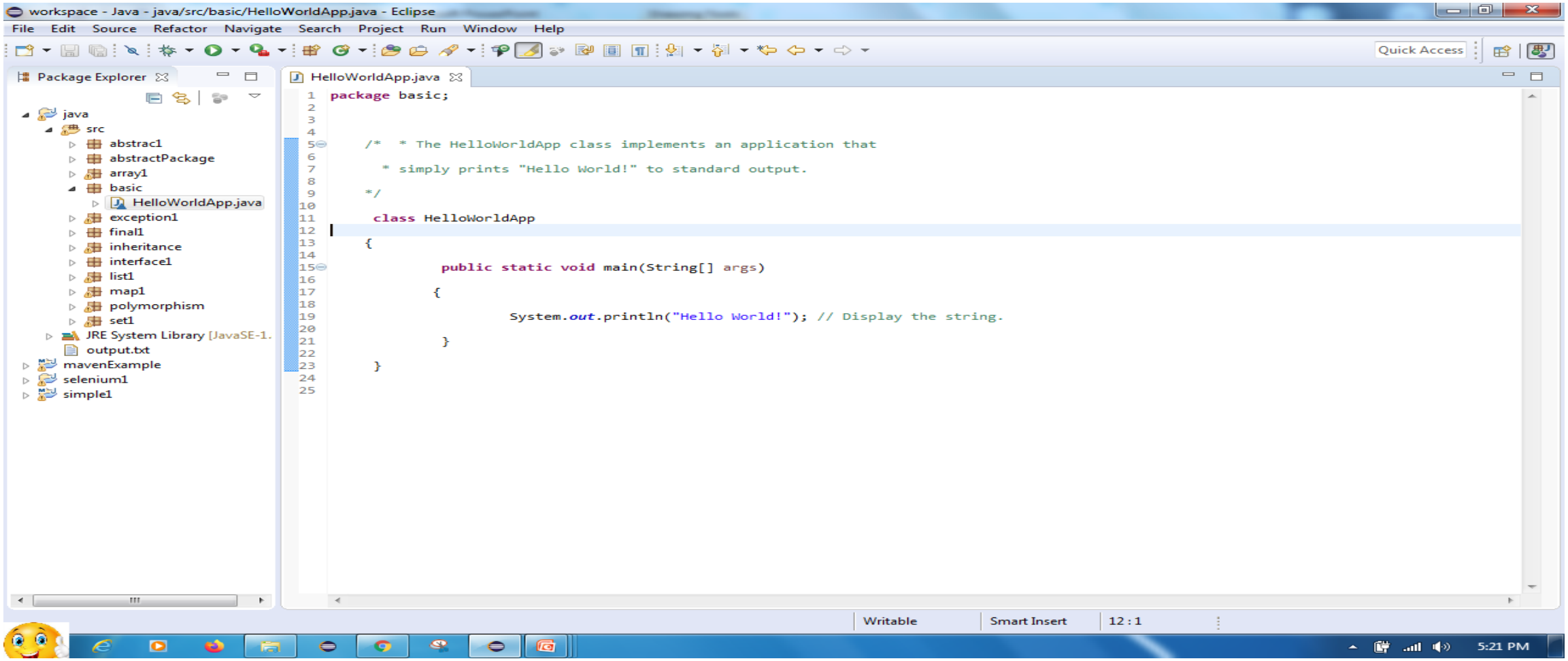
Eclipse IDE

Start working by creating the Java Projects



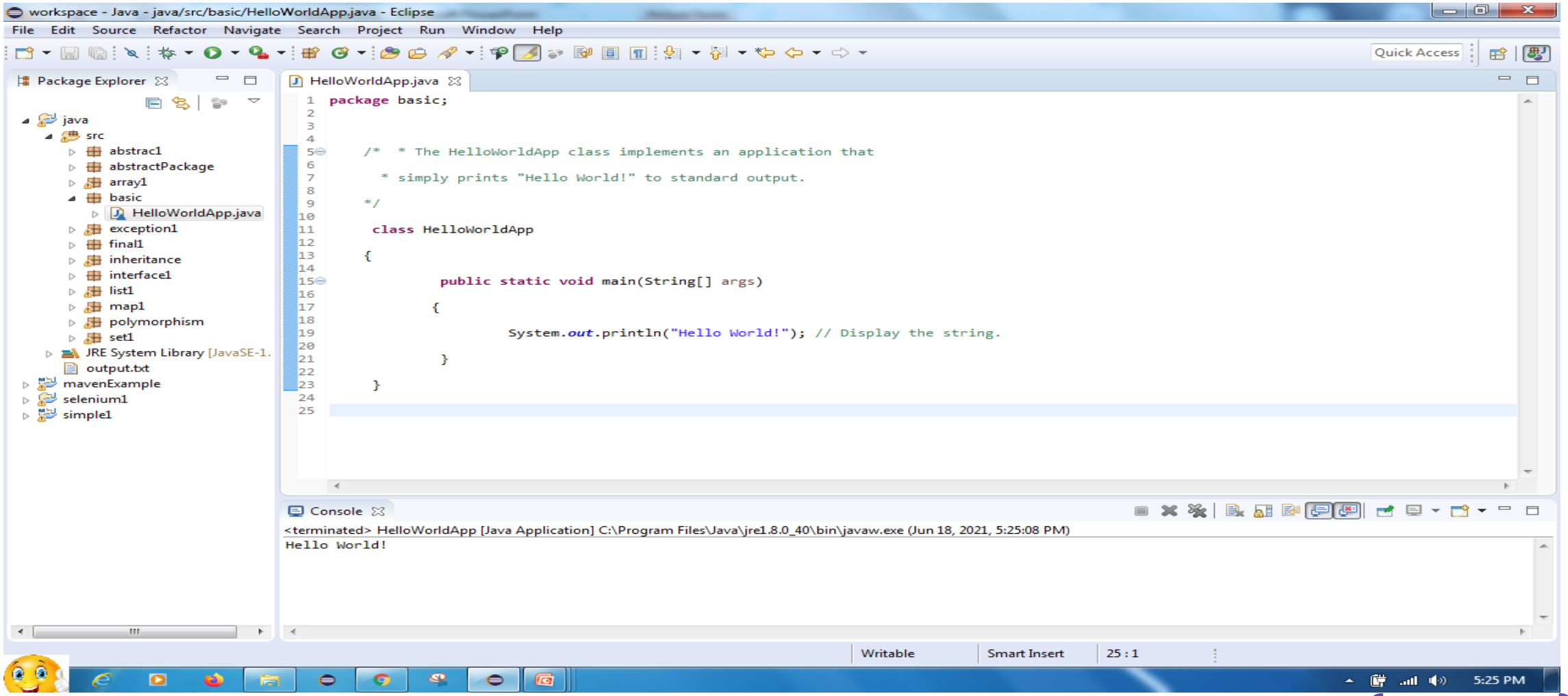
Eclipse IDE

Create First Java Program with file extension .java Example: HelloworldApp.java



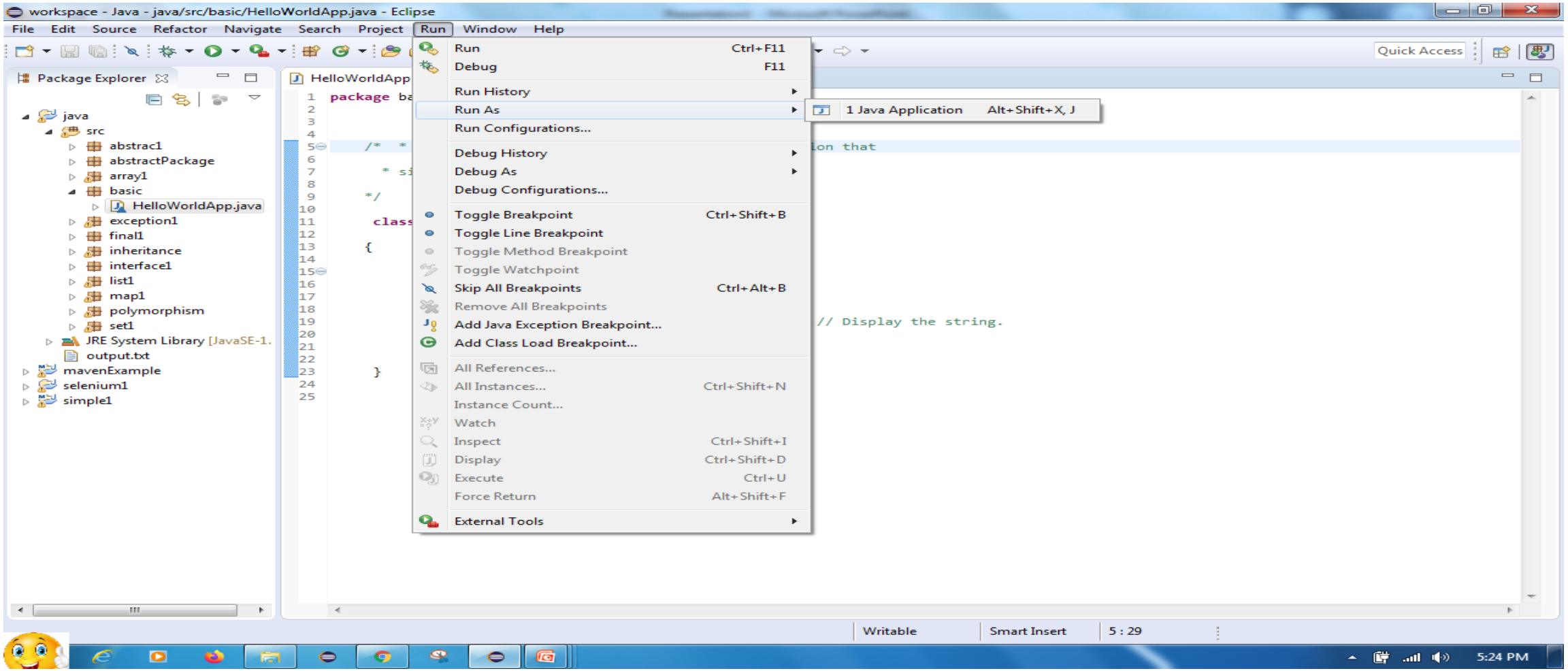
Eclipse IDE

After run Print Hello World!



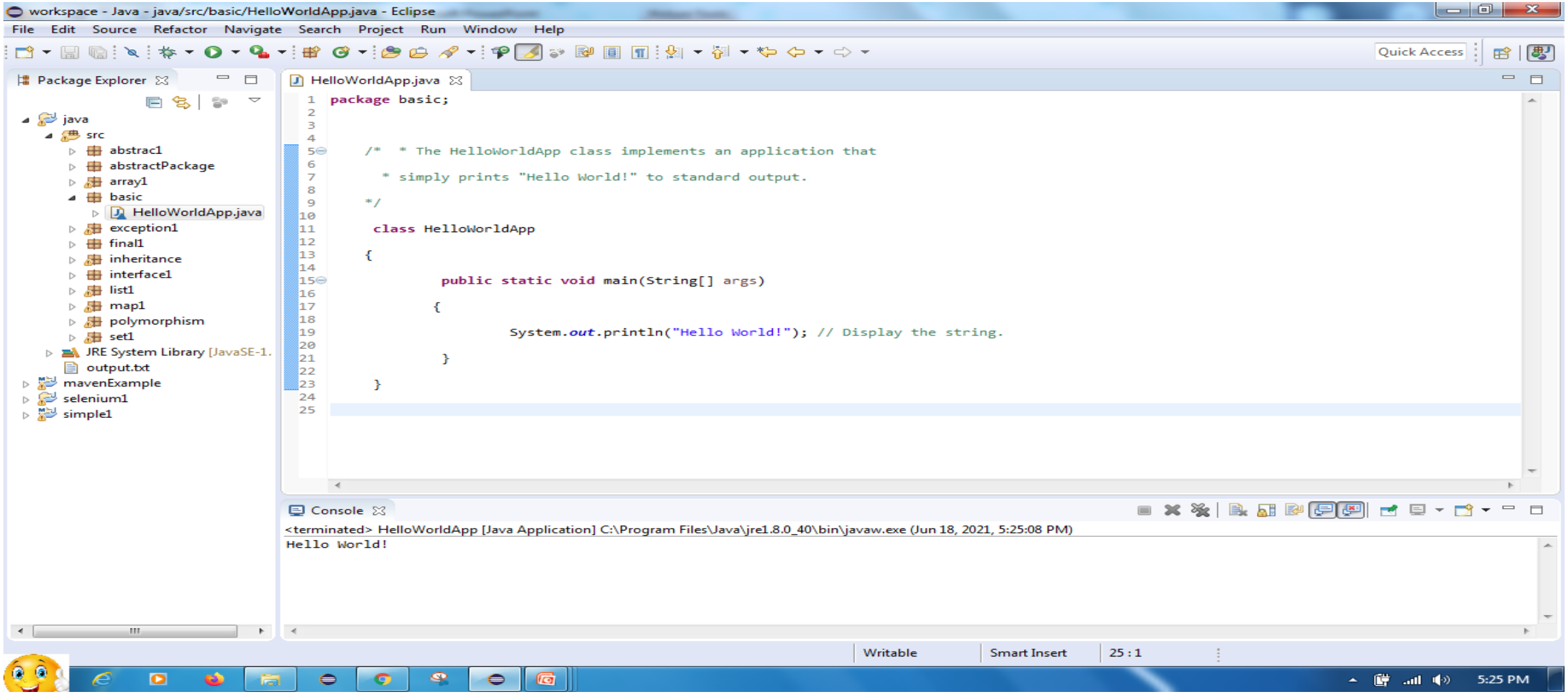
Eclipse IDE

Create First Java Program with file extension .java Example: HelloworldApp.java



Eclipse IDE

After run Print Hello World!



First Java Program

```
/**
 * The HelloWorldApp class implements an application that
 * simply prints "Hello World!" to standard output.
*/
class HelloWorldApp {
        public static void main(String[] args){
                System.out.println("Hello World!"); // Display the Hello World!
        }
}
```

Need of Coding standards

- Code conventions are important to the programmer for the following reasons:

  - To facilitate the **changing, and maintenance** of the code.

  - To maintain the **readability** of the code.

  - Helps to **understand the code quickly** and thoroughly.

  - Helps to **well package the code** when deploying as the product.

Java source files

- Source file of java can be saved with the extension or suffixes **“Filename.java”.**
- **Java source file has the following order**:
    - Comments
    - Package and import statements
    - Class and interface declaration

Comments

- Comments are the **overview** or **description** of the code.
- In Java, the comments can be implemented in **four styles** as follows
- **Block comments:**
  - ✓ Block comments **(/***/)** are used to provide **descriptions of files, methods, data structures, and algorithms.**
  - ✓ Block comments may be used at the **beginning** of each file and before each method and also within methods.
  - ✓ Block comments **inside** a function or method should be indented to the **same level** as the code they describe.

Example:

```
/*
*Here is a block comment
*/
```

Comments

- **Single-Line comments:**
  - ✓ **Short description of a block of code** can be given using single-line comments **(/*..*/).**
  - ✓ If a comment can't be written in a single line, it should follow the block comment format.

Example:

```
if(condition)  {
    /*   Description about bock of code in a single line */
                    }
```

Comments

- **End - Of - Line comments:**
  - ✓ This comment line **(//)** can be used as a **single-line comment** or short comment for a statement.
  - ✓ These comments can be included in the **same line** of the statement but should be **separated** from the statement using **spaces**.
  - ✓ This can be used in **consecutive multiple lines** for commenting out sections of

Example:

```
class HelloWorldApp     {
        public static void main(String[] args)   {
                System.out.println("Hello World!");      // Display the Hello World!
        }
 }
```

Documentation Comments

- Documentation comments are generally used when **writing code for a project/software package**.
- It helps to generate a **documentation page for reference**, which can be used for getting information about the present method, its parameters, class, variables, etc..,
- The **JavaDoc** tool is used to process the doc comments that come with JDK and it is used for generating Java code documentation in HTML(HyperText Markup Language) format from Java source code, which requires documentation in a predefined format.
- It is made up of **two parts**: **a description and Javadoc tags.**

Documentation Comments

Javadoc Tag:

- **Special tag** embedded within a Java doc comment.
- These doc tags enable you **to autogenerate** a complete, **well-formatted API from your source code.**
- The tags **start** with an "at" sign (@) and are **case-sensitive** (i.e).,they must be in upper and lowercase letters as shown in the below table.
- A tag must start at the beginning of a line (after any leading spaces and an optional asterisk) or it is treated as normal text.
- By convention, tags with the **same name are grouped together**.

First Java Program using Documentation Comments

```
/**
 * The HelloWorldApp class implements an application that
 * simply prints "Hello World!" to standard output.
 @author SmartCliff
*/
class HelloWorldApp    {
        public static void main(String[] args)    {
                System.out.println("Hello World!");      // Display the Hello World!
        }
}
```

Doc comment

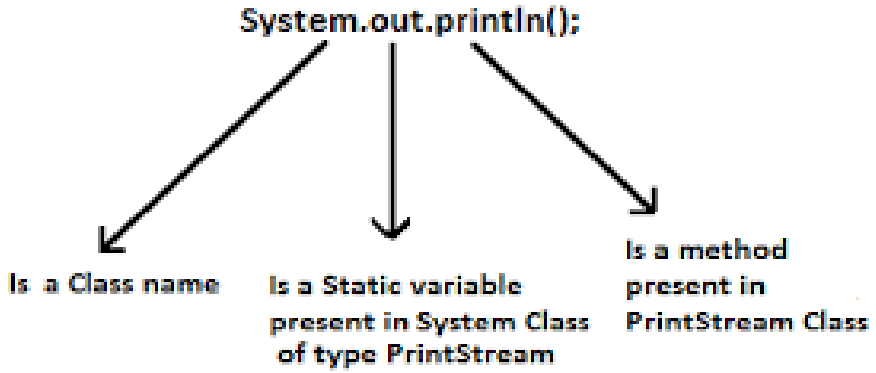
First Java Program in Detail

- **Class:** keyword is used to declare a class in java.
- **public** keyword is an access modifier which represents visibility, it means it is visible to all.
- **static** is a keyword, if we declare any method as static, it is known as static method.
- The core **advantage of static method** is that there is **no need to create object** to invoke the static method.
- The **main method is executed by the JVM**, so it doesn't require to create object to invoke the main method. So it saves memory.
- **void** is the return type of the method, it means it doesn't return any value.

First Java Program in Detail

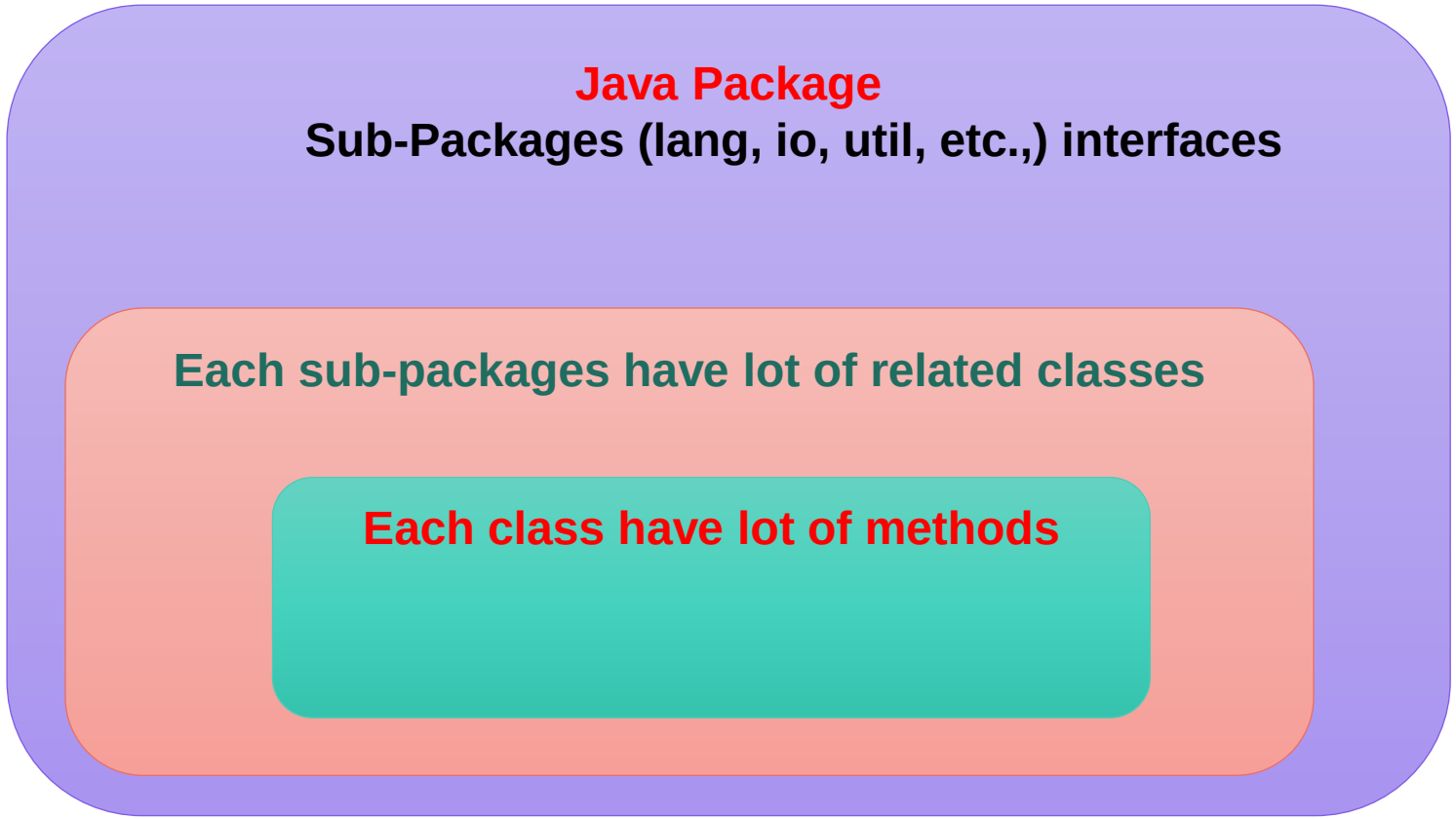
- **main** represents startup of the program.
- **String[] args** is used for command line argument.
- **System.out.println()** is used print statement.

  - System is a final class defined in the **java.lang package.**



Package: Introduction

- **Package:** It is a mechanism to **organize related classes, interfaces, and sub-packages** according to their functionality. It is like a folders in a file directory.



Package: Introduction

There are two types of Packages:

1. **Built-in Packages:** The already defined package in Java API like **java.io.*, java.lang.*** etc. are known as built-in packages. It is simply **import** based on your application needs.

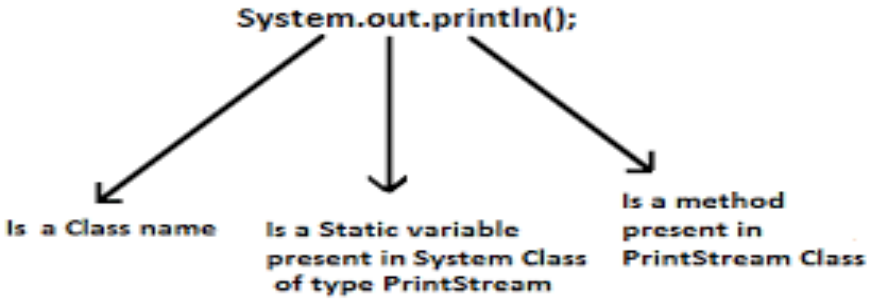
2. **User-defined Packages:** The package created by user and use based on application needs is called user-defined package.

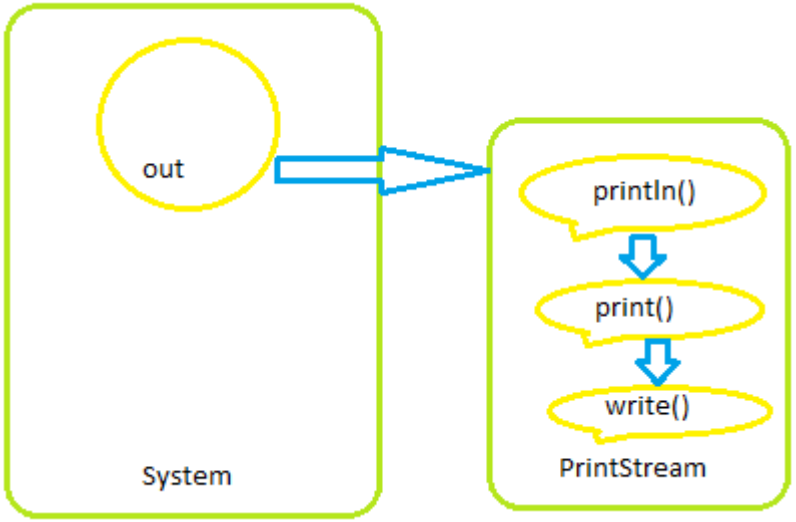
Note:

- Programmers typically **use packages** to **organize classes** belonging to the **same category** or providing **similar functionality**.

Package: Introduction

- **Example**: java.lang.System (System class is a one of the class of lang package)





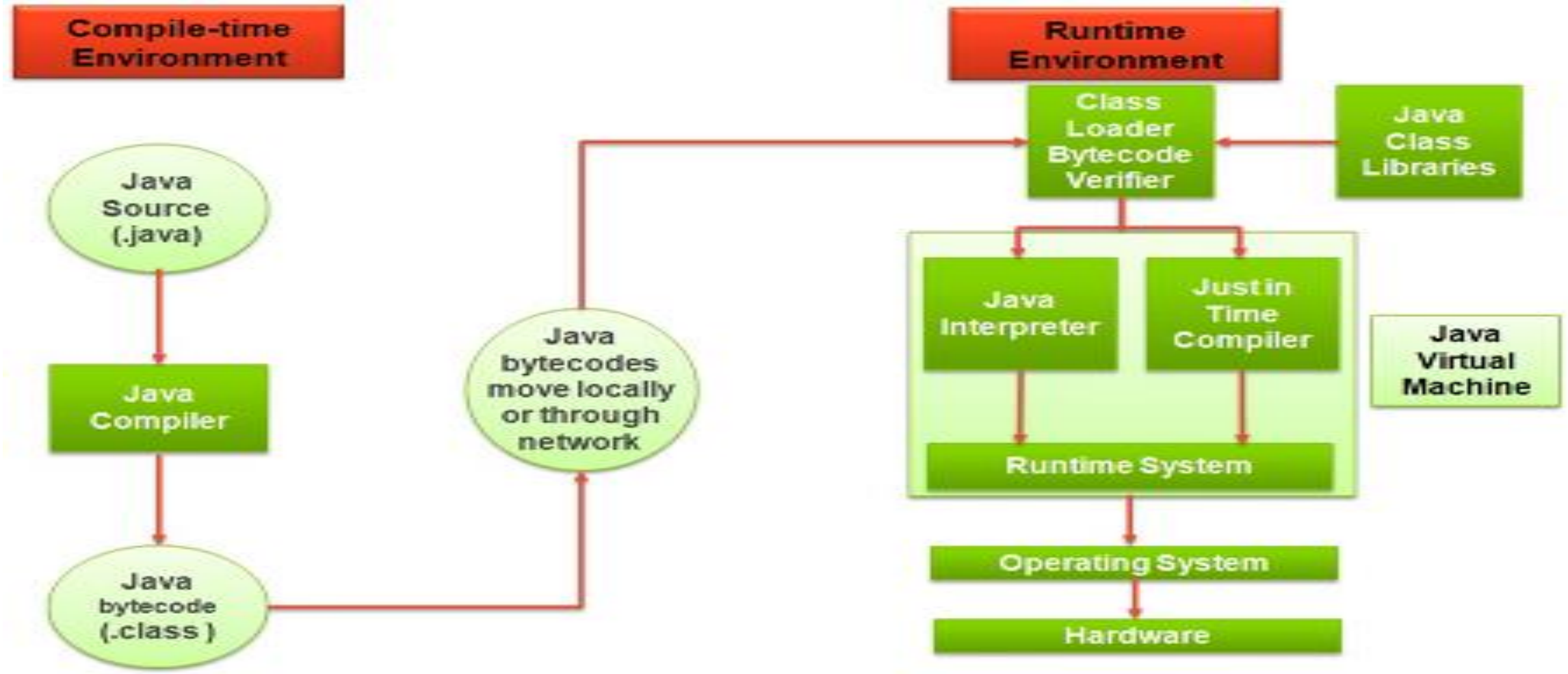
Note:

- As per Java 1.8 standard version, Java have **14 predefined packages**, **150 sub packages**, **7000 classes** and **7 lakh methods**.

Compiling and Executing

- To run java application both **compiler and interpreter** involves.
- The source code of Java will be created with file extension .java, **Example**: HelloworldApp.java
- The java Compiler compiles a java file and convert into **bytecode / class file**.
- The byte code will be in a file with extension .class
- The .class file is **interpreted** by the JVM and morphed into **machine specific code**.

Compiling and Executing



Just-In-Time (JIT) Compiler

- The **Just-In-Time (JIT) compiler** is one of the integral parts of the Java Runtime Environment.
- It improves the **performance of Java applications** by compiling byte codes to native machine code at run time.
- The JIT compiler is **enabled by default**. When a method has been compiled, the JVM calls the compiled code of that method directly instead of interpreting it.
- Theoretically, if compilation did not require processor time and memory usage, compiling every method could allow **the speed of the Java program** to approach that of a native application.

Java Tokens

Introduction

- The tokens are **the small building blocks** of a Java program that are meaningful to the Java compiler.
- The Java **compiler breaks the line** of code into text (words) is called **Java tokens**.
- The Java compiler identified these **words as tokens**.
- These tokens are **separated** by the **delimiters** and delimiters are not part of the Java tokens.
- It is useful for **compilers to detect errors.**

Example: In program, we will be using many statements and expressions to perform the operation. These statements and expressions are made up of tokens.

Introduction

- Java supports **5 types of tokens** as follows:

    1. Keywords
    2. Identifiers
    3. Literals
    4. Operators
    5. Special symbols

Keywords

- Keywords **are predefined or reserved words** that have special meaning to the Java compiler.
- Each keyword is assigned a **special task** or function and cannot be changed by the user.
- We **cannot** use keywords as **variables or identifiers** as they are a part of Java syntax itself.
- A keyword should always be written in **lowercase** as Java is a case-sensitive language.

Keywords

- They are **some available keywords** in Java

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

Identifiers

- An identifier is the **name given by the user** for the **various programming elements like variables,classes, methods, interface, etc.**
- **Rules for defining Identifiers**
  - Allowed characters for identifiers are all **alphanumeric characters**([A-Z],[a-z],[0-9]), '$'(dollar sign) and '_' (underscore).
    - **Example**: "blue@" is not a valid as it contain '@' special character.
  - Identifiers should not start with digits([0-9]).
    - **Example:** "123blue" is a not a valid
  - Java identifiers are **case-sensitive.**
    - **Example:** test and Test both are different

Identifiers

- There is no limit on the length of the identifier, but it is advisable to use an optimum length of **4 – 15 letters** only.
- Reserved Words can’t be used as an identifier.
  - **Example**: int while = 20; is an **invalid statement** as **while is a reserved word**.
- **Some Valid Identifiers**

  test, Test, _name, name100

  ABC, File_100, _abc_

Some Invalid Identifiers

int, 100name, test@123

Test.txt,

Naming Convention of Identifiers

- All the **identifiers** such as **classes, interfaces, packages, methods, and fields** of Java programming language are given according to the Java **naming convention.**
- By using this naming convention, we can achieve **readability** and can also easily **understand the code**.
- In programming, we often **remove the spaces** between words because programs of different sorts reserve the space (‘ ’) character for special purposes.
- Because the **space character is reserved**, we cannot use it to represent a concept that we express in our human language with multiple word

Naming Convention of Identifiers

Example

- That is in programming we cannot refer, *user login count=5,*represent as userLoginCount = 5.

- Java follows the **CamelCase** for identifiers naming conventions.

CamelCase

- **Combines the compound words** and removes the space between the words.

- **Two types**: UpperCamelCase and lowerCamelCase.

- **UpperCamelCase:** The first letter of each word is capitalized.

  Example: Product, ProductDescription, CountValue.

- **lowerCamelCase :** The first letter of the compound words is lowercase.

  Example: product, iPad, countValue, productDescription

Naming Convention : Variables

- A variable's name can be **any legal identifier** i.e., begins with a letter, the dollar sign "$", or the underscore character "_".
- The variable name should be **short** and **meaningful.**
- The choice of a variable name should be **mnemonic**- that is, designed to indicate to the casual observer the intent of its use.
- **One-character variable** names should be **avoided** except for temporary variables.
- The **temporary variables** can be **i,j,k,m, and n for integer** and **c,d,e for characters.**
- The variable name should be in **lowerCamelCase.**

Example:

```
String firstName;
int orderNumber ;
int count;
float price;
```

Naming Convention : Variables

- **Private class variables** should have an **underscore prefix**.
- Apart from its name and its type, the scope of a variable is its most important feature.
- Indicating class scope by using underscore makes it easy to **distinguish class variables** from local scratch variables.
- This is important because class variables are considered to have higher significance than method variables, and should be treated with special care by the programmer

Example:

```
class Login{
private String _userName;
...
 }
```

Naming Convention : Constants and Methods

Constants:

- The names of variables declared class constants and should be in **uppercase**.
- If we specify the constant with **two words**, then separated by **underscores ("_").**

Example:

```
static final int MAX_HEIGHT = 70;
static final int MAX_WIDTH = 100;
static final int LENGTH = 20;
```

Naming Convention : Constants and Methods

Methods

- Method should be in the verb.
- Method name should be **lowerCamelCase.**

Example:

void calculateTax()

String getSurname()

void draw()

Naming Convention : Package and Import statements

- The first non-comment line of the java source file is a **package followed by import statements.**
- The prefix of a unique package name is always written in **all-lowercase ASCII letters**.

Example:

```
package java.util;
import java.util.Scanner;
```

Naming Convention in Class and interface

- **Class names** should be **nouns** and must be **simple** and **descriptive.**
- Use whole words-**avoid acronyms** and **abbreviations.**
- Class name should be in **UpperCamelCase.**

Example:

class Customer

class CustomerAccount

class Login

Naming Convention in Class and interface

- **Interface** tends to have a name that describes an operation that a class can do.
- Name of the interface should be **UpperCamelCase.**

Example:

interface Enumerable

interface CompEnumerable

interface Login

Data Types

- Data types defines the type data a variable can hold. It specify the different sizes and values that can be stored in the variable.
- Java is a **strongly typed language**
  - All variables must be declared before its use.
- **Two Types of Data types**

1. **Primitive data type**
   - byte, short, int, long, float, double, char, boolean
2. **Non Primitive data type**
   - Classes, interfaces, arrays, strings

Data Types

- **byte: (1 byte)**:
  - –The byte data type is an 8-bit signed two's complement integer.
  - –It has a minimum value of -128 and a maximum value of 127 (inclusive).
  - –Useful for saving memory in large arrays.
  - –**Default value** : 0
- **short: (2byte)**
  - –The short data type is a 16-bit signed two's complement integer.
  - –It has a minimum value of -32,768 and a maximum value of 32,767 (inclusive).As byte ,can use a short to save memory in large arrays
  - –**Default value** : 0

Data Types

- **int: (4 byte)**
  - By default, the int data type is a 32-bit signed two's complement integer, which has a minimum value of -2^{31} and a maximum value of $2^{31} -1$.
  - **Default value** : 0
- **long**: **(8 byte)**
  - The long data type is a 64-bit two's complement integer.
  - The signed long has a minimum value of -2^{63} and a maximum value of $2^{63}-1$
  - **Default value** : 0L
- **float: (4 byte)**
  - The float data type is a single-precision 32-bit IEEE 754 floating point.
  - **Default value** : 0.0f

Data Types

- **double: (8 byte)**
  - The double data type is a double-precision 64-bit IEEE 754 floating point
  - **Default value** : 0.0d
- **boolean: (1 bit)**
  - The boolean data type has only two possible values: true and false
  - **Default value** : false
- **char**: **(2 byte)**
  - single 16-bit **Unicode** character. It has a minimum value of '\u0000' (or 0) and a maximum value of '\uffff'
  - **Default value** : '\u0000'

Variables

- A **symbolic name associated with a value** and whose associated value may be changed.
- A variable is defined by the combination of an **identifier, a type and an optional initializer.**
- All variables have a **scope, which defines their visibility, and a lifetime**.
- There are **three types** of variables**:**

    1. Local Variables
    2. Instance Variables (Non-Static Fields)
    3. Class Variables (Static Fields)

Variables

Local Variables

- Variable that's declared within the body of a method.
- Will be used only within that method.
- Other methods in the class aren't even aware that the variable exists.
- A method will often store its state in local variables.
- A local variable **cannot** be defined with **"static"** keyword.

Variables

Instance Variables (Non-Static Variables)

- A variable declared **inside the class but outside the body of the method**, is called instance variable.
- It is **not declared as static**.
- Instance variables are created when the objects are instantiated and therefore they are **associated with the objects.**

Variables

Class Variables (Static Variables) :

- A variable which is **declared as static is called static variable**. It is also called as a **class variable**
- It **cannot be local**.
- You can create a single copy of static variable and share among all the instances of the class.
- Memory allocation for static variable **happens only once** when the class is loaded in the memory.

Variables

```
/**
  * The VariableApp class implements an application that
  * illustrate different Java variable
*/
 class VariableApp {
            int mark = 95 ;//instance variable
            static char grade = 'S';     // static variable
            public static void main(String[] args) {
                        float average=95.0  // local variable
            }
 }
```

Literals

- A **fixed value** assigned to a variable

- Different **types of literals** are as follows:

Types	Examples

Integer Literals
1. Decimal
2. Hexa Decimal
3. Binary
	int decValue=56;
	int hexaValue=0x10;
	int binVal=0b11010 ;

Floating-Point Literals	double d1 = 123.4;
	double d2 = 1.234e2;
	float f1 = 123.4f;

Character Literals	char chrlit='8';
String Literals	String check="Test" ;
Boolean Literals	boolean result=true;

Literals

- **Using Underscore Characters in Numeric Literals**
- In **Java SE 7 and later**, any number of **underscore characters** (_) can **appear anywhere** between digits in a numerical literal.
- Enables **to separate groups of digits in numeric literals**, which can improve the readability of your code.

Example

```
long creditCardNumber = 1234_5678_9012_3456L;

long socialSecurityNumber = 999_99_9999L;

long bytes = 0b11010010_01101001_10010100_10010010;
```

Literals

- You can **place underscores** only between digits; you **cannot place underscores** in the following places:
  - At the beginning or end of a number
  - Adjacent to a decimal point in a floating point literal
  - Prior to an F or L suffix
  - In positions where a string of digits is expected

Unicode

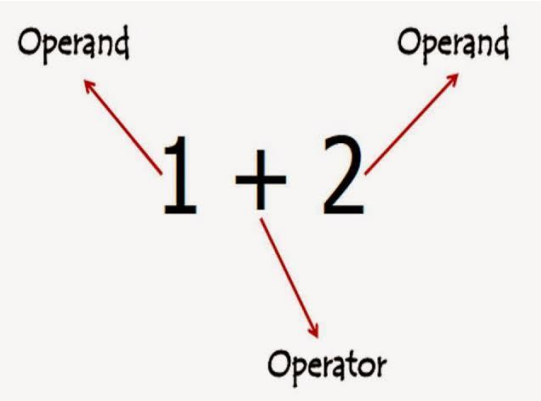
- It is Computing industry standard designed to **encode characters of the world's written languages**.
- **Unicode System?**
  - Before Unicode, there were many language standards:
  - ASCII (American Standard Code for Information Interchange) for the United States.
  - ISO 8859-1 for Western European Language.
  - KOI-8 for Russian.
  - GB18030 and BIG-5 for Chinese, and so on.

Unicode

- **Problems:**
  - –A particular code value corresponds to different letters in the **various language standards**.
  - –The encodings for languages with large character sets have variable length.
  - – Some common characters are encoded as single bytes, other require two or more byte.
- **Solution:**
  - –To solve these problems, a new language standard was developed i.e. Unicode System.
  - –In **Unicode, character holds 2 byte**, so java also uses 2 byte for characters.
  - –The **range of a char is 0 to 65,536**

Operators

- An operator in Java is a **special symbol** that signifies the **compiler** to perform **some specific mathematical or non-mathematical operations on one or more operands**.
- Value that the operator operates on is called operand.



- Java supports **8 types of operators**.

Expression

- An expression in Java is **any valid combination of tokens** like variables, constants and operators.
- An expression may consist of **one or more operands, and zero or more operators** to produce a value.

Examples:

- a+b*c
- (a*b)/(c+d)
- 10-4*5
- Etc.,

Operators & Expressions

- Below table we have listed down all the operators, along with their expressions:

Type	Operators	Expressions
Unary Operator	++,--,+(unary),-(unary), ~,!	a++,--a, -a, ~a, !a
Arithmetic Operator	+,-,%,*,/	a=b+c, a=b%d
Shift Operator	<<,>>,>>>	a=a>>2
Relational Operator	>,>=,<,<=,==, !=,instanceof	a=10>5,5!=10,8==8
Bitwise Operator	&,^,|	a=2&3
Logical Operator	&&,||	(2>5)&&(10>8)
Ternary Operator	? :	a=(b>c)?1:0
Assignment Operator	= ,+=, -=, *= ,/= ,%= ,&= ,^= ,|= <<= ,>>=, >>>=	a=b a+=b a^=b

Precedence and Associativity

Precedence

- Operator precedence determines the **order** in which the operators in an expression are evaluated**.**

Example: int num = 6 -3 *8;

- To evaluate the above expression Java, consider the precedence of the operator .Here, Multiplication (*) has highest precedence than subtraction (-).So multiplication will be performed before the subtraction.

Precedence and Associativity

Associativity

- If an **expression has two operators** with **similar precedence,** the expression is evaluated according to its **associativity .**
- That is either **left to right**, or **right to left.**

Example: X=Y=Z; Operator has same precedence. Here, the value of Z is assigned to variable Y. Then the value of Y is assigned of variable X because the associativity of = operator is from right to left

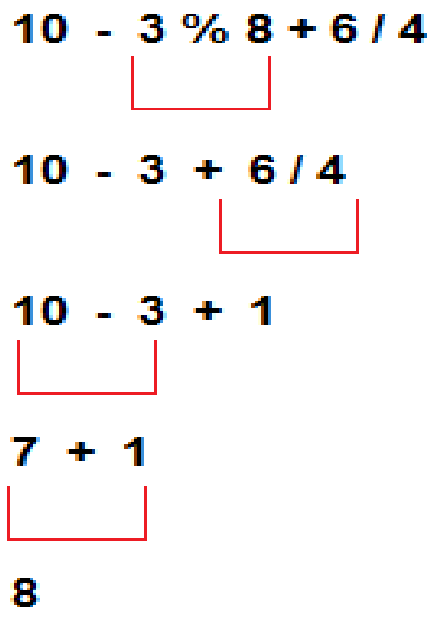
Precedence and Associativity

Operators	Precedence	Associativity
postfix increment and decrement	++ --	left to right
prefix increment and decrement, and unary	++ -- + - ~ !	right to left
multiplicative	* / %	left to right
additive	+ -	left to right
shift	<< >> >>>	left to right
relational	< > <= >= instanceof	left to right
equality	== !=	left to right

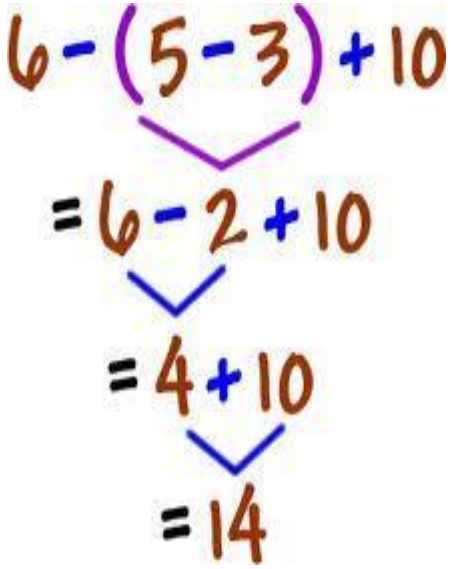
Precedence and Associativity

Operators	Precedence	Associativity
bitwise AND	&	left to right
bitwise exclusive OR	^	left to right
bitwise inclusive OR	|	left to right
logical AND	&&	left to right
logical OR	||	left to right
ternary	? :	right to left
assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=	right to left

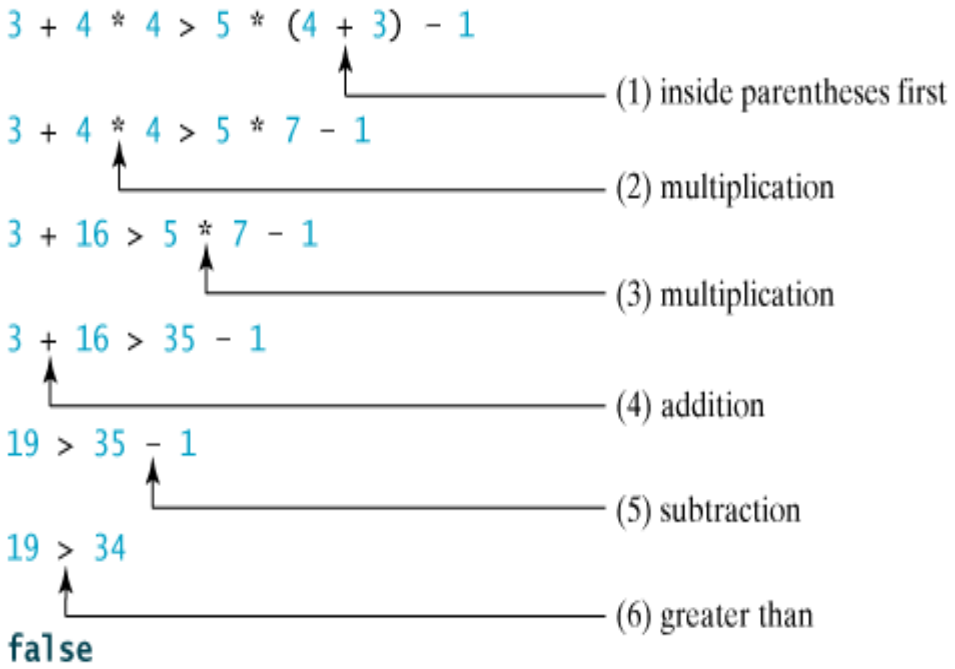
Expression Evaluation



Example 1



Example 2



Example 3

Type Conversions

- **Widening or Automatic Type Conversion –** lower data types are automatically convert into higher data types. It is also known as **implicit conversion**.
- Automatic type conversion will take place if the following **two conditions** are met:
  - –The two types are compatible.
  - –The destination type is larger than the source type

Type Conversions

- Widening primitive conversions:

  byte → short, int, long, float, double

  short → int, long, float, double

  char → int, long, float, double

  int → long, float, double

  long → float, double

  float → double

- **Generally safe** because they tend to go from a small data type to a larger one
- No automatic conversion is supported from **numeric type to char or boolean**

Type Conversions

```
/**
 * The ConversionAutomatic class implements an application that
 * Illustrate the automatic type conversion
*/
 class ConversionAutomatic {
        public static void main(String[] args) {
                        int i = 100;
                        long l = i; // automatic type conversion
                        float f = l; // automatic type conversion
                        System.out.println("Int value "+i);
                        System.out.println("Long value "+l);
                        System.out.println("Float value "+f);
        }
 }
```

Type Conversions

- **Narrowing or Explicit Conversion -** If we want to assign a value of **larger data type to a smaller data type** we perform explicit type casting or narrowing.
- Useful for incompatible data types **where automatic conversion cannot be done**.
- Here, target-type specifies the desired type to convert the specified value to.
- **Narrowing primitive conversions**:

byte → char
short → byte, char
char → byte, short
int → byte, short, char
long → byte, short, char, int
float → byte, short, char, int, long
double→ byte, short, char, int, long, float

Type Conversions

```
/**
  * The ConversionExplicit class implements an application that
  * Illustrate the explicit type conversion
*/
 class ConversionExplicit {
        public static void main(String[] args)  {
                        double d = 100.04;
                        long l = (long)d;   //convert double into long
                        int i = (int)l;  // long convert into int
                        System.out.println("Double value "+d);
                        System.out.println("Long value "+l);
                        System.out.println("Int value "+i);
        }
 }
```

Type Conversions

- **Type promotion in Expressions -** While evaluating expressions, the intermediate value may **exceed the range of operands** and hence the expression value will be promoted.
- **Some conditions for type promotion are:**

1. Java automatically promotes each byte, short, or char operand to int when evaluating an expression.
2. If one operand is a long, float or double the whole expression is promoted to long, float or double respectively.

Type Conversions

```
/**
 * The TypePromotion class implements an application that
 * Illustrate the type promotion  */

class TypePromotion{
      public static void main(String[] args){
                  byte b = 50;
                  b = (byte)(b * 2);   //promote into int
                  System.out.println(b);
      }
}
```

Type Conversions

```
/**
 * The TypePromotion1 class implements an application that
 * Illustrate the type promotion  */

class TypePromotion1{
	public static void main(String[] args){
		byte b = 42;
		char c = 'a';
		short s = 1024;
		int i = 50000;
		float f = 5.67f;
		double d = .1234;
		double result = (f * b) + (i / c) - (d * s); //promote into double
		System.out.println("result = " + result);
	}
}
```

Special Symbols

- Special symbols in Java are a **few characters which have special meaning** known to Java compiler and cannot be used for any other purpose.
- In the below table we have listed down the special symbols supported in Java along with their description.

Special Symbols

Symbols	Description
brackets []	These are used as an array element reference and also indicates single and multidimensional subscripts
Parentheses()	These indicate a function call along with function parameters
Braces{}	The opening and ending curly braces indicate the beginning and end of a block of code having more than one statement
Comma (,)	This helps in separating more than one statement in an expression
Semi-Colon (;)	This is used to invoke an initialization list

Read User Input

Introduction

- There are **three different ways** to read input from user:
  1. Buffered Reader Class
  2. Console Class
  3. Scanner Class
- **Scanner Class**: It is a class in **java.util package** used for obtaining the user input of the primitive types like int, double and strings. It is the **easiest way to read input** in a Java program.
- **Scanner object** is constructed from **Scanner Class** and **System.in (input stream)** object is passed as a parameter while creating a scanner object.
- **Constructing a Scanner object to read console input:**

```
Scanner scannerObject = new Scanner(System.in);
```

Methods

- After creating the scanner object, we can use below **Scanner class methods** for reading the **respective primitive data types from the console.**

Method	Description
boolean nextBoolean()	This method reads the boolean value from the user.
byte nextByte()	It reads the byte value from the user.
double nextDouble()	It accepts the input in double datatype from the user.
float nextFloat()	It takes the float value from the user.
int nextInt()	It reads the integer value from the user.

Methods

Method	Description
String nextLine()	This method reads the String value from the user.
long nextLong()	This method reads the long type of value from the user.
short nextShort()	It reads the short type of value from the user.
String next()	It reads the String input from the user.

Methods

Note:

- Scanner class **no specific method** to read character type.
- To read a single character, we use **next().charAt(0). next() function** returns the next token/word in the **input as a string** and **charAt(0) funtion** returns the **first character** in that string.

Example: #1

```
/**
  * The ReadSomeInput class implements an application that
  * Illustrate reading a console input  */
import java.util.Scanner; //import Scanner class from util package
public class ReadSomeInput {
        public static void main(String[] args) {
                Scanner console = new Scanner(System.in);
                System.out.print("Enter your Name : ");
                String name = console.next();
                System.out.println("Hi, " +name+" . Welcome to the Training
Program ...");
                console.close();
    }
}
```

Output:

```
Enter your Name : Arun
Hi, Arun . Welcome to the Training
Program ...
```

Example: #2

```
/**
  * The ReadInput class implements an application that
  * Illustrate reading a console input
*/
import java.util.Scanner; //import Scanner class from util package
public class ReadInput {
public static void main(String[] args) {
        Scanner console = new Scanner(System.in);
        System.out.print("Enter your User Name : ");
        String name = console.next();
        System.out.print("Enter your Password : ");
        String password = console.next();
```

Example: #2

```
        if(name == "Admin" && password == "admin@123") {
            System.out.println("You are Successfully Logged in");
        }
        else {
            System.out.println("User Name or Password is Wrong");
        }
        console.close();
    }
}
```

Output1:

Enter User Name : Admin

Enter Password : admin@123

You are Successfully Logged in

Output2:

Enter User Name : Admin

Enter Password : admin

User Name or Password is Wrong

Example: #3

```
/**
  * The ReadMovie class implements an application that
  * Illustrate reading a different types of input
*/
import java.util.Scanner;
public class ReadMovie {
public static void main(String[] args) throws ParseException {
        Scanner console = new Scanner(System.in);
        System.out.println("Enter Movie ID : ");
        int movieid = console.nextInt();
        System.out.println("Enter Movie Name : ");
        String moviename = console.next();
        System.out.println("Enter Movie Description : ");
        String moviedescription = console.next();
        System.out.println("Enter Movie Language : ");
        String movielanguage = console.next();
        System.out.println("Enter Movie Genre : ");
```

Example: #3

```
      String moviegenre = console.next();
       System.out.println("Enter Movie release date (dd/mm/yyyy) : ");
       String date = console.next();
       System.out.println("Enter Movie Seat Cost : ");
       float movieseatcost = console.nextFloat();
       System.out.println("ENTERED MOVIE DETAILS ARE");
       System.out.println("Movie ID : "+movieid);
       System.out.println("Movie Name : "+moviename);
       System.out.println("Movie Description : "+moviedescription);
       System.out.println("Movie Language : "+movielanguage);
       System.out.println("Movie Genre : "+moviegenre);
       System.out.println("Movie Date : "+moviedate);
       System.out.println("Movie Seat Cost : "+movieseatcost);
} }
```

Output:

```
Enter Movie ID :  1231
Enter Movie Name : AAA
Enter Movie Description : Dramaof1956
Enter Movie Language : English
Enter Movie Genre : ACTION
Enter Movie release date (dd/mm/yyyy) :  23/03/2022
Enter Movie Seat Cost :  120.0
ENTERED MOVIE DETAILS ARE
Movie ID : 1231
Movie Name : AAA
Movie Description : Dramaof1956
Movie Language : English
Movie Genre : ACTION
Movie Date : 23/03/2022
Movie Seat Cost : 120.0
```

Quiz

1) Which of the following are primitive data types?

a) int

b) float

c) double

d) boolean

e) All the above

e)All the Above

Quiz

2) A local variable stores temporary state; it is declared inside a

- a) Class
- b) Method
- c) Block
- d) Object

- b) & c)

Quiz

3) A _____ is a value that should not be altered by the program during normal execution.

a) Variable

b) int

c) Identifiers

d) Constant

d) Constant

Quiz

4) Which of these can not be used for a variable name in Java?

- a) Identifier
- b) Keyword
- c) Both a and b
- d) None of the above

b) Keyword

Quiz

5) Which conversion also called Automatic Type Conversion?

a) Widening

b) Narrowing

c)Both A and B

d) All the above

a) Widening

Quiz

6) Long Literals in java must be appended by which of these?

- a) L
- b) l
- c) D
- d) 0x

- a) & b)

Quiz

7) Which variables are created when an object is created and destroyed when the object is destroyed?

a) Local variables

b) Instance variables

c) Class Variables

d) Static variables

b) Instance variables

Quiz

8) Which of the following are not Java keyword ?

a) double

b) switch

c) instanceof

d) then

d) then

Quiz

9) Which of these is not a bitwise operator?

a) &

b) |

c) ^

d) <=

d) <=

Quiz

10) Which operator is used to invert all the digits in binary representation of a number?

a) ~

b) <<

c) ^

d) >>>

a) ~

Quiz

11) It is time to buy a new phone when at least one of the following situations occurs:

- **the phone breaks**
- **the phone is at least 3 years old**

```
int phoneAge;        // in years
boolean isBroken;
……..                 //code initializes variables
boolean needPhone =_____________
```

```
(isBroken == true) || (phoneAge >= 3);
```

Quiz

12) Evaluate the following expression:

(17 < 4*3+5) || (8*2 == 4*4) && !(3+3 == 6)

a) true

b) false

b)false

Quiz

13) Assume num1=10. Which of the following is not a valid statement?

a) num1>>2

b) num1<<<2

c) num1%=2;

d) num1<<2;

b) num1<<<2

Quiz

14) In Java, after executing the following code what are the values of x, y and z?

```
int x,y=10; z=12;
x=y++ + z++;
```

a) x=22, y=10, z=12

b) **x=24, y=10, z=12**

c) x=24, y=11, z=13

d) x=22, y=11, z=13

d) x=22, y=11, z=13

Quiz

15) Which Scanner class method is used to read integer value from the user?

a)next()

b) nextInt()

c) nextInteger()

d) readInt()

b) nextInt()

Quiz

16) Which is the correct syntax to declare Scanner class object?

a) Scanner obj=Scanner();

b) Scanner obj=new Scanner();

c) Scanner obj= Scanner (System.in);

d) Scanner obj=new Scanner (System.in);

d) Scanner obj=new Scanner (System.in);

Quiz

17) Consider the following object declaration statement

Scanner obj= new Scanner(System.in).

What is System.in in this declaration?

a)Class which point input device

b) Reference to Input stream

c) Reference to Computer System

d)None of these

b) Reference to Input stream