**JAVA AS OBJECT-ORIENTED PROGRAMMING**

**Prepared by fredy**

**COMMAND PROMPT(CMD)**

**Is the tools in window that allows you to type commands directly to control your computer without using mouse.**

**Class**

* A **class** is a **blueprint(ramani)** or **template** for creating objects.
* Class ni mfano wa **object(kitu)** ila siyo **object(kitu).**
* It defines **properties** (attributes)(data members/variables) and **methods(function)** (behaviors of an object).
* It **cannot be used directly** without creating an object you can’t access.

**Example of a Class**

**Java code**

**// Class: Defines the blueprint for a Carclass Car {**

**String name;**

**int year;**

***// Method***

**void honk() {**

**System.out.println("Beeep! Beeep!");**

**}**

**}**

**Object**

* An **object** is an **instance of a class** (a real example of the blueprint).
* Object ni mfano halisia wa class furani.
* It represents a real-world entity that can perform actions.
* You can create **multiple objects** from a single class.

**Example of an Object.**

**Java code**

**public class Main {**

**public static void main(String[] args) {**

***// Creating an object from the Car class***

**Car car1 = new Car();**

**car1.name = "Toyota";**

**car1.year = 2020;**

***// Using a method from the class***

**car1.honk();**

***// Displaying object properties***

**System.out.println("Name: " + car1.name);**

**System.out.println("Year: " + car1.year);**

**}**

**}**

DIFFERENCE BETWEEN COMPILATION AND EXECUTION

Compilation

is the process of converting human-readable code into Machine-readable code.

Performed by Java compiler(javac) to generate **bytecode** with file extension .class;

\***bytecode**

is the intermediate machine -independent code that is generated after a program is compiled.

After generation of bytecode **java virtual Machine (jvm)** in java it convert bytecode (form .class files) into Machine code that the OS can understand.

Bytecode allows java program to be “write once, Run anywhere” (WORA) because it is the same to all platform where a JVM is available.

Execution

Is the process of runs the machine-readable code to perform the intended operations. It is performed by **CPU** and **Operating System** that leads to produces result according to program’s logic.

**EXPLANATION ABOUT SYSTEM.OUT.PRINTLN();**

**System**

Is a **built-in class** that provides useful tools for interacting with computer system. It help task like-:

1.printing message/output using tools (**System.out**)

2.Reading input using tools (**System.in**)

3.Handling error or error messages (**System.err**)

Therefore **in, out, exit and err** are object within class System:

**DIFFERENCE BETWEEN JVM ,JDK AND JRE**

**\*\*JDK\*\* (Java Development Kit)**

**is the software that provides everything needed to develop, compile, and run java program. Consist of JRE+Compiler(Javac)+Debuging Tools.**

**\*\*JRE\*\* (Java Runtime Environment)**

**Refers to the package that includes JVM and Essential Java Libraries.**

**Allows users to run java application but does not include tools for writing or compiling java code.**

**\*\*JVM\*\* (Java Virtual Machine)**

**Refers to non-physical components but a runtime engine that executes java programs.**

**Convert** Bytecode (**from .class file**) into **Machine code that the OS and CPU can understand.**

**Analogy:**

**Think of java as a car**

**JVM The engine that runs the car**

**JRE The car itself, ready to drive**

**JDK A factory with all tools to built and run a car**

**JDK**

**Compiler JRE**

**Class file(.class)**

**Bytecode JVM**

**OUTPUT**

Extension during compilation and run

**.java Compilation .class(bytecode) JVM=OUTPUT**

**USER INPUT IN JAVA**

**There are two ways of inputing**

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

**SCANNER**

**Refers to the predefined class which is used to take user input in java defined inside Java.util** Package.

**Importing the package**:

import java.util.scanner;

\*\*import

is used in java to bring in a tools or class from another place (a library) so you can use it in your program.

Is like to say “Hey, I want to use this tools , so bring it to my program”

\*\*Java.util

Java is a big box that holds many tools for programming.

Util is a smaller box inside that big box that has tools like **Scanner, ArrayList. etc.**

**\*\*Scanner**

**Is a tools(class) inside java. util that help you to read what the user types on the keyboard.**

**Used for input of outside things from the user.**

**IN SIMPLE**

**Import** getting tools to use

**Java.util** is where the tools is located

**Scanner** is the tools used to get input from the user

**SUMMARY OF ITS METHOD**

1. nextLine(): Reads a full line of text
2. next(): Reads a single word(token)
3. nextInt(): Reads an Integers
4. nextDouble(): Reads a decimal number
5. nextBoolean(): Reads a Boolean value
6. hasNext(): Checks if there’s more input to read
7. hasNextLine(): Checks if there’s more input(line)
8. skip(): Skip unwanted input(e.g., Whitespace)
9. close(): Closes the Scanner.

**BUFFEREDREADER**

**Is the class used to read text from character-based input stream, like the keyboard or/ file, more efficiently than scan.**

**It take large amounts of data in larger chunks(Buffer) to make reading faster**

**It reads only text(String) unlike Scanner read all types of data:**

**It belongs within :-**

**Java.io** libraries;

Summary of key methods:

1. readLine(): Reads a full lines of a text
2. read(): Reads a single character
3. skip(): Skips a number of characters
4. mark() and reset(): marks a position and resets to it
5. close(): Closes the reader and releases resourses.

**HOW TO INPUT DATA FROM KEYBOARD**

**Why is needed to create an object to read input from keyboard**

1. **import. util. Scanner;** is not enough to read input on its own ,it only imports the Scanner class ,but you still to create a scanner object in your program to actually read input.

**CREATING OBJECT IN SCANNER**

**Scanner input1=new Scanner(System.in);**

* **Scanner**

**Is the class name from java’s built-in java.util packages**

* **Input1**

**Is the object name (or variable name) you can give any name of your choice**

* **New**

**Is the keyword used to create a new object in java.**

* **Scanner(…)**

**Is the constructor of the Scanner class**

* **System.in**

**Used to read input from the keyboard.**

**FULL EXPLANATION**

* **we create an object input1 of the scanner class.**
* **The new keyword creates the Scanner object in memory**
* **System.in tells the java to read user input from the keyboard**
* **Now we can use input1 to read the values like numbers, words etc.**

**STATIC(Class property) KEYWORD IN JAVA:**

**it means that something belongs to the class(ujumla) itself rather than instance(kitu kimoja)(object) of a class.**

**\*\*Normally when we create objects from class each objects gets its own copy of instance variables or methods.**

**But when we use static, it means that the variables, methods is shared among all instance of a class, rather than to be unique to each other:**

**How static used in java:**

* **Static variables:**

**Variables shared by all objects of a class**

* **Static method:**

A method that can be called without creating an object

* **Static block:**

Code that runs once when the class is loaded

* **Static Class:**

A nested class that can be used without an instance of the outer class

STATIC

It is used for memory managements and serving space of memory.

**[OOP]**

**Explanation about java as object-oriented programming**

**(Program itumiayo objects or instances)**

**(OOP)**

**Is a way of writing program where we use objects(vitu) instead of just functions and variables.**

**Real-life example:**

**Imagine your building a program to manage cars in a transportation company.**

**\*\*A car has properties like color, speed, and brands;**

**\*\* A car also has behaviors like starting, Stopping and accelerating**

**Instead of writing code of each instance (cars) separate, we create a blueprint (called class) and use it to make different cars(object).**

**Class will be a car**

**Object will be type of car like honda, Toyota e.t.c;**

**MAIN CONCEPT OF OOP**

**Class and objects**

* **classes**

**is the template for creating object (like a car blueprint)**

* **Objects is an actual instance created from a class**

**ENCAPSULATION(kuficha)**

**Is the process of hiding(kuficha) data to prevent unauthorized access. keeping data private using object within class;**

**Eg. Is like to store money in a bank and securing it with a PIN. You can’t withdraw money more than your balance**

**Code**

**Private double balance;**

**ABSTRACTION (Kuficha vitu vigumu na kuonesha vitu muhimu)**

**Is the process of hiding unnecessary details and showing only important parts.**

**Eg.ATM: You select “Withdraw Money” but you don’t see how the bank transfers the money internally.**

**Code**

**abstract class vehicle{**

**abstract void move();**

**User can work with objects without needing to understand internal mechanisms.**

**INHERITANCE**

**Is the process of allowing one class to inherit properties and methods from another class.**

**Family: A child can inherit eyes color, face shape, or talents from their parents.**

**In java we use extends to allow a child class to inherit from a parent class.**

**// parent class**

**Class Animal{**

**Void eat(){**

**System.out.println(“ this animal is eating,…..”);**

**}**

**}**

**// child class(inherits animal)**

**Class Dog extends Animal{**

**Void bark(){**

**System.out.println(“dog is barking : woof! Woof!”);**

**}**

**}**

**POLYMORPHISM(ONE ACTION, MANY FORMS)**

**Is the process of allowing the same method to work in different ways.**

**Is like smartphone -you can use for calling, taking photos, or online but still the same device.**

**Eg: Different animals make different sounds, but they all fall under the “animal” category.**

**Class Animal{**

**Void makeSound(){**

**System.out.println(“this animal makes sound…..”);**

**}**

**}**

**// child class(inherits animal)**

**Class Dog extends Animal{**

**Void makeSound(){**

**System.out.println(“dog is barking: woof! Woof!”);**

**}**

**}**

**Class Cat extends Animal{**

**@override**

**Void makeSound(){**

**System.out.println(“Cat is meows: Meow! Meow!”);**

**}**

**}**

**Public class TestPolymorphism{**

**Public static void main(String[] args){**

**Animal myAnimal=new Dog();**

**myAnimal.makeSound();**

**myAnimal=new Cat();**

**myAnimal.makeSound();**

**}**

**}**

**In generally;**

**Polymorphism it makes the programs flexible and easy to modify.**

**Jump statement in java**

**Are used to control the flow of program execution by skipping(kuruka) or terminating(stopping) certain parts of code.**

**There are three main jump statements**

1. **Break**
2. **Continue**
3. **Return**

**BREAK;**

**The break statements immediately stop the execution of a loop or switch case.**

**Code**

**Public class BreakExample{**

**Public static void main(Sring[] args){**

**for(int i=1; i<=5; i++){**

**if(i==3){**

**break;**

**}**

**System.out.println(“Number: “+i);**

**}**

**}**

**}**

**For loop starts with i=1 and runs up to i=5.**

**When i==3, break stops the loop, so 3 and the following numbers are not printed**

**OUTPUT:**

**Number:1**

**Number:2**

**CONTINUE**

**(skipping current iteration)**

**Is used to skip the current iteration without stopping the loop:**

**Public class ContinueExample{**

**Public static void main(String[] args){**

**Int i=0;**

**while(i<=5){**

**i++;**

**if(i==3){**

**continue;*// skips iteration when i==3***

**}**

**System.out.println(“Number: “+i);**

**}**

**}**

**}**

**\*\*The number 3 is skipped but the loop continue**

**OUTPUT:**

**Number:1**

**Number:2**

**Number:4**

**Number:5**

**RETURN STATEMENT**

**It stops method execution and can return a value if the method has a return type.**

**Public class ReturnExample{**

**Public static void main(String[] args){**

**Int result=square(4);**

**System.out.println(“Result: “+result);**

**}**

**Public static int square(int num){**

**Return num\*num;**

**}**

**}**

**OUTPUT**

**Result:16**

**\*\*WHILE LOOP\*\***

**(wakati hali Fulani ni kweli):**

**While(condition){**

**// code to be excuted**

* **Condition-A Boolean expression(true or false)**
* **The loop continue execution-**as long as condition is true
* **If condition become false-**the loop terminates immediately.

**\*\*Do While loop\*\***

**“fanya kwanza, kisha angalia kama hali bado ni kweli ili kurudia”.**

**Do{**

***// tekeleza msimbo huu***

**} while(condition);**