Java and OOPs Notes

Knowledge Gate (Golu Sir)

About these Notes

These notes are based on the tutorial from our Youtube Video

https://www.youtube.com/watch?v=LepLT82RtxA

Basics

Java is one of the most popular programming languages in the world. With Java you can build various types of applications such as desktop, web, mobile apps and distributed systems.

Java Development Kit

We use Java Development Kit (JDK) to build Java applications. JDK contains a compiler, the Java Runtime Environment (JRE) and a library of classes that we use to build applications.

How Java Code Gets Executed

The Java compiler takes Java code and compiles it down to Java Bytecode which is a cross-platform format. When we run Java applications, Java Virtual Machine (JVM) gets loaded in the memory. It takes our bytecode as the input and translates it to the native code for the underlying operating system. There are various implementations of Java Virtual Machine for almost all operating systems.

Architecture of Java Applications

The smallest building blocks in Java programs are **methods** (also called functions in other programming languages). We combine related methods in **classes**, and related classes in **packages**. This modularity in Java allows us to break down large programs into smaller building blocks that are easier to understand and re-use.

5 Amazing Facts about Java

1. Java was developed by James Gosling in 1995 at Sun Microsystems (later acquired by Oracle).
2. It was initially called Oak. Later it was renamed to Green and was finally renamed to Java inspired by Java coffee.
3. Java has close to 1 Crore developers worldwide.
4. About 1300 crore devices run java.
5. According to indeed.com, the average salary of a Java developer is just over $100,000 per year in the US.

Data Types

Variables

We use variables to temporarily store data in computer's memory. In Java, the type of a variable should be specified at the time of declaration.

In Java, we have two categories of types:

- **Primitives:** for storing simple values like numbers, strings and booleans.
- **Reference Types:** for storing complex objects like email messages.

Primitive Types

Type	Bytes	Range
byte	1	[-128, 127]
short	2	[-32K, 32K]
int	4	[-2B, 2B]
long	8	
float	4	
double	8	
char	2	A, B, C, …

boolean	1	true / false

Declaring Variables

```
byte age = 30;
long viewsCount =
3_123_456L; float price
= 10.99F;
char letter = 'A';
boolean isEligible = true;
```

- In Java, we terminate statements with a semicolon.
- We enclose characters with single quotes and strings (series of characters) with double quotes.
- The default integer type in Java is int. To represent a long value, we should add L to it as a postfix.
- The default floating-point type in Java is double. To represent a float, we should append F to it as a postfix.

Comments

We use comments to add notes to our code.

```
// This is a comment Sanchit sir is the best
```

Reference Types

In Java we have 8 primitive types. All the other types are reference types. These types don't store the actual objects in memory. They store the reference (or the address of) an object in memory.

To use reference types, we need to allocate memory using the new operator. The memory gets automatically released when no longer used.

```
Date now = new Date();
```

Strings

Strings are reference types but we don't need to use the new operator to allocate memory to them. We can declare string variables like the primitives since we use them a lot.

```
String name = "Sanchit";
```

Useful String Methods

The String class in Java provides a number of useful methods:

- startsWith("a")
- endsWith("a")
- length()
- indexOf("a")
- replace("a", "b")
- toUpperCase()
- toLowerCase()

Strings are immutable, which means once we initialize them, their value cannot be changed. All methods that modify a string (like toUpperCase) return a new string object. The original string remains unaffected.

Escape Sequences

If you need to use a backslash or a double quotation mark in a string, you need to prefix it with a backslash. This is called escaping.

Common escape sequences:

- \\
- \”
- \n (new line)
- \t (tab)

Arrays

We use arrays to store a list of objects. We can store any type of object in an array (primitive or reference type). All items (also called elements) in an array have the same type.

```
// Creating and and initializing an array of 5
elements int[] numbers = new int[3];
numbers[0] = 10;
numbers[1] = 20;
numbers[2] = 30;

// Shortcut
int[] numbers = { 10, 20, 30 };
```

Java arrays have a fixed length (size). You cannot add or remove new items once you instantiate an array. If you need to add new items or remove existing items, you need to use one of the collection classes.

The Array Class

The Array class provides a few useful methods for working with arrays.

```
int[] numbers = { 4, 2,
7 };
Arrays.sort(numbers);
String result =
Arrays.toString(numbers);
System.out.println(result);
```

Multi-dimensional Arrays

```
// Creating a 2x3 array (two rows, three
columns) int[2][3] matrix = new int[2][3];
matrix[0][0] = 10;
```

```
// Shortcut int[2][3] matrix = {
```

```
};
```

```
{ 1, 2, 3 },
{ 4, 5, 6 }
```

Constants

Constants (also called final variables) have a fixed value. Once we set them, we cannot change them.

```
final float INTEREST_RATE = 0.04;
```

By convention, we use CAPITAL LETTERS to name constants. Multiple words can be separated using an underscore.

Arithmetic Expressions

```
int x = 10 + 3;

int x = 10 - 3;
int x = 10 * 3;
int x = 10 / 3;                    // returns an
int float x = (float)10 / (float)3; // returns
a float int x = 10 % 3;     // modulus
(remainder of division)
```

Increment and Decrement Operators

```
int x = 1;
x++;    // Equivalent to x =
x + 1 x--;   // Equivalent to
x = x - 1
```

Augmented Assignment Operator

```
int x = 1;
x += 5; // Equivalent to x = x + 5
```

Order of Operations

Multiplication and division operators have a higher order than addition and subtraction. They get applied first. We can always change the order using parentheses.

```
int x = 10 + 3 * 2;    // 16
int x = (10 + 3) * 2;  // 26
```

Casting

In Java, we have two types of casting:

- **Implicit**: happens automatically when we store a value in a larger or more precise data type.
- **Explicit**: we do it manually.

```
// Implicit casting happens because we try to store a short
// value (2 bytes) in an int (4
bytes). short x = 1;
int y = x;

// Explicit
casting int x
= 1;
short y = (short) x;
```

To convert a string to a number, we use one of the following methods:

- Byte.parseByte(“1”)
- Short.parseShort(“1”)

- Integer.parseInt(“1”)
- Long.parseLong(“1”)
- Float.parseFloat(“1.1”)
- Double.parseDouble(“1.1”)

Formatting Numbers

```
NumberFormat currency =
NumberFormat.getCurrencyInstance(); String result =
currency.format(“123456”); // $123,456
```

```
NumberFormat percent =
NumberFormat.getPercentInstance(); String result =
percent(“0.04”); // 4%
```

Reading Input

```
Scanner scanner = new
Scanner(system.in); double number =
scanner.nextDouble(); byte number =
scanner.nextByte();
String name = scanner.next();
String line =
scanner.nextLine();
```

Control Flow

Comparison Operators

We use comparison operators to compare values.

```
x == y  // equality operator
x != y.   // in-equality
operator x > y
x
>
=
y
x
<
y
x
<
=
y
```

Logical Operators

We use logical operators to combine multiple boolean values/expressions.

- x && y (AND): if both x and y are true, the result will be true.
- x || y (OR): if either x or y or both are true, the result will be true.
- !x (NOT): reverses a boolean value. True becomes false.

If Statements

Here is the basic structure of an if statement. If you want to execute multiple statements, you need to wrap them in curly braces.

The Ternary Operator

```
String className = (income > 100_000) ? "First" : "Economy";
```

Switch Statements

We use switch statements to execute different parts of the code depending on the value of a variable.

After each **case** clause, we use the break statements to jump out of the switch block.

For Loops

For loops are useful when we know ahead of time how many times we want to repeat something. We declare a loop variable (or loop counter) and in each iteration we increment it until we reach the number of times we want to execute some code.

```
    for (int i = 0; i < 5; i++) statement
```

While Loops

While loops are useful when we don't know ahead of time how many times we want to repeat something. This may be dependent on the values at run-time (eg what the user enters).

```
while (someCondition) {
```

```
    …
        if (someCondition) break;
}
```

We use the **break** statement to jump out of a loop.

Do..While Loops

Do..While loops are very similar to **while** loops but they executed at least once. In contrast, a while loop may never get executed if the condition is initially false.

```
do {
    …
} while (someCondition);
```

For-each Loops

For-each loops are useful for iterating over an array or a collection.

```
int[] numbers = {1, 2, 3, 4}; for (int number
: numbers)

    …
```

Access Modifiers:

Defines access type of the method i.e. from where it can be accessed in your application. In Java, there 4 type of the access specifiers.
public: accessible in all class in your application.
protected: accessible within the package in which it is defined and in its subclass(es)(including subclasses declared outside the package)
private: accessible only within the class in which it is defined.
default (declared/defined without using any modifier): accessible within same class and package within which its class is defined.

OOPS Concepts

Object

Any entity that has state and behavior is known as an object. For example, a chair, pen, table, keyboard, bike, etc. It can be physical or logical.

An Object can be defined as an instance of a class. An object contains an address and takes up some space in memory. Objects can communicate without knowing the details of each other's data or code. The only necessary thing is the type of message accepted and the type of response returned by the objects.

Example: A dog is an object because it has states like color, name, breed, etc. as well as behaviors like wagging the tail, barking, eating, etc.

Class

Collection of objects is called class. It is a logical entity.

A class can also be defined as a blueprint from which you can create an individual object. Class doesn't consume any space.

Inheritance

When one object acquires all the properties and behaviors of a parent object, it is known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.

Polymorphism

If *one task is performed in different ways*, it is known as polymorphism. For example: to convince the customer differently, to draw something, for example, shape, triangle, rectangle, etc.

In Java, we use method overloading and method overriding to achieve polymorphism.

Another example can be to speak something; for example, a cat speaks meow, dog barks woof, etc.

Abstraction

Hiding internal details and showing functionality is known as abstraction. For example phone call, we don't know the internal processing.

In Java, we use abstract class and interface to achieve abstraction.

Encapsulation

Binding (or wrapping) code and data together into a single unit are known as encapsulation. For example, a capsule, it is wrapped with different medicines.

A java class is the example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here.

Coupling

Coupling refers to the knowledge or information or dependency of another class. It arises when classes are aware of each other. If a class has the details information of another class, there is strong coupling. In Java, we use private, protected, and public modifiers to display the visibility level of a class, method, and field. You can use interfaces for the weaker coupling because there is no concrete implementation.

Cohesion

Cohesion refers to the level of a component which performs a single well-defined task. A single well-defined task is done by a highly cohesive method. The weakly cohesive method will split the task into separate parts. The java.io package is a highly cohesive package because it has I/O related classes and interface. However, the java.util package is a weakly cohesive package because it has unrelated classes and interfaces.

Association

Association represents the relationship between the objects. Here, one object can be associated with one object or many objects. There can be four types of association between the objects:

- One to One
- One to Many

- Many to One, and
- Many to Many

Let's understand the relationship with real-time examples. For example, One country can have one prime minister (one to one), and a prime minister can have many ministers (one to many). Also, many MP's can have one prime minister (many to one), and many ministers can have many departments (many to many).

Aggregation

Aggregation is a way to achieve Association. Aggregation represents the relationship where one object contains other objects as a part of its state. It represents the weak relationship between objects. It is also termed as a *has-a* relationship in Java. Like, inheritance represents the *is-a* relationship. It is another way to reuse objects.

Composition

The composition is also a way to achieve Association. The composition represents the relationship where one object contains other objects as a part of its state. There is a strong relationship between the containing object and the dependent object. It is the state where containing objects do not have an independent existence. If you delete the parent object, all the child objects will be deleted automatically.