Date Jun 1 (2025)
Page No.

# 0. Introduction To Programming

## (a) Types of Languages (Programming paradigm)

### (i) Procedural programming

- Step-by-step instructions, like a recipe.

⇒ A programming style where code is written as a sequence of step-by-step instructions using functions and procedure

### ii) Functional programming

- Use pure functions and avoid changing data.

⇒ A way of programming using functions that don't change data and don't affect anything outside (no side effects).

### (iii) Object-Oriented Programming (OOP)

- Organize code around real-world objects.

⇒ A programming style that organizes codes into objects, which contain data (fields) and behaviour (methods).

## Static VS Dynamic Languages

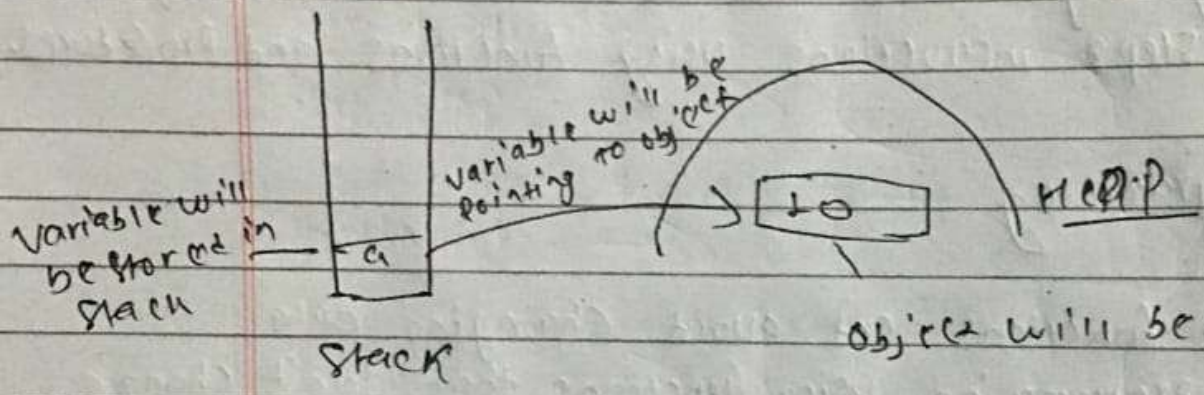
| Static | Dynamic |
|---|---|
| i) Perform type checking at compile time. | i) Perform type checking at runtime. |
| ii) Errors will show at compile time. | ii) Error might not show till programs is run. |
| iii) Declare data type before you use it. | iii) No need to declare datatype of variables. |
| Ex- int a = 10;<br>String name = "ravi";<br>(needs to declare datatypes) | Ex- a = 10;<br>name = "ravi"<br>(no need to declare datatypes) |

(0) Works in both static & dynamic programming

b. How variables and objects are stored.

a = 10 -> Object

Reference variable




Object will be stored in Heap.

- More than one reference variable can point to the same object.
- A change in one variable will be applied in all variables point the object.

Ex-

```
int[] a = [1, 3, 5, 9]
    b = a
  a[0] = 99
```

-> this makes [1,3,5,9] to [99,3,5,9]

```
System.out.println(b);
```

output will be -> [99, 3, 5, 9].

(a)

Date
Page No.

## Garbage collection (static)

Objects that doesn't have reference variable ~~it will~~ they will be removed when garbage collection is hited.

Ex: a = 10. → [a] → (10) Heap

a = 37 → Here it shifts from stack 10, 10 → 37.

[a] → [37]

- So, 10 is left without reference variable
- Now, garbage collection will come & remove this 10.

- This will not be done in dynamic programming