QB MODULE 1

1. Features of Java Language

Java has several features, also called **buzzwords** (Page 56 & Page 143 of PDF):

- **Simple** – Easy to learn, uses familiar C/C++ syntax but removes complexities like pointers.
- **Secure** – Provides runtime checking, avoids explicit pointer use, and supports bytecode verification.
- **Object-Oriented** – Everything in Java is based on objects and classes.
- **Robust** – Strong memory management, exception handling, and garbage collection.
- **Multithreaded** – Supports multiple tasks running simultaneously.
- **Architecture-Neutral & Portable** – Java bytecode can run on any platform with JVM.
- **Interpreted & High Performance** – Compiled into bytecode and executed efficiently.
- **Distributed** – Built-in support for networking and sharing data across machines.

2. Principles of Object-Oriented Programming

The four **OOP principles** (Page 46–52):

1. **Abstraction** – Hiding complex implementation details and exposing only essential features. Example: A car is driven without knowing its engine mechanism.
2. **Encapsulation** – Wrapping data (variables) and methods (functions) together inside a class, restricting direct access. Example: Private variables with public methods.
3. **Inheritance** – One class (child) acquires properties and methods of another class (parent) using extends. This allows code reusability.
4. **Polymorphism** – Ability of one method to perform different tasks depending on context. Achieved by **method overloading (compile-time)** and **method overriding (runtime)**.

3. Constructor Method with Example

Found on (Page 40–41):

- A **constructor** is a special method that initializes objects when they are created.
- It has the **same name as the class**, does not return any value, and is called automatically.

Example (Parameterized Constructor):

```
class Student {
   int id; String name;
   Student(int i, String n) {   // constructor
```

```
    id = i; name = n;
  }
  void display() {
    System.out.println(id + " " + name);
  }
  public static void main(String[] args) {
    Student s1 = new Student(101, "Ravi");
    Student s2 = new Student(102, "Neha");
    s1.display(); s2.display();
  }
}
```

4. Control Statement in Java

Found on (Page 61–70):

- **Control statements** are used to change the flow of execution. They are:

1. **Selection Statements** → if, if-else, if-else-if, nested if, switch.
2. **Iteration Statements (Loops)** → for, while, do-while.
3. **Jump Statements** → break, continue, return.

Example:

```
if(age > 18) {
  System.out.println("Eligible to vote");
} else {
  System.out.println("Not eligible");
}
```

5. Inheritance Property to Display Book Information in Java

Found on (Page 47–49):

- **Inheritance** allows one class to acquire the properties and methods of another using extends.
- The child class (subclass) inherits variables and methods of the parent class (superclass).

Example:

```
class Book {
  String title, author;
  void setBook(String t, String a) {
    title = t; author = a;
  }
  void display() {
    System.out.println(title + " by " + author);
  }
}
class Novel extends Book {
  int price;
  void setPrice(int p) { price = p; }
  void show() {
    display();
    System.out.println("Price: " + price);
  }
  public static void main(String[] args) {
    Novel n = new Novel();
    n.setBook("Java Basics", "James Gosling");
    n.setPrice(500);
    n.show();
  }
}
```

6. Looping Statement in Java

Found on (Page 83–90):

- **Loops** are used to execute a block of code repeatedly.

1. **for loop** – Best when the number of iterations is known.
2. for(int i=1;i<=5;i++) { System.out.println(i); }
3. **while loop** – Executes as long as the condition is true.
4. int i=1; while(i<=5){ System.out.println(i); i++; }

5. **do-while loop** – Executes at least once before checking the condition.

```
int i=1; do { System.out.println(i); i++; } while(i<=5);
```

7. Java Program to Calculate Simple Interest

Found on (Page 31 – Exercises):

Formula: SI = (P × R × T) / 100

```
import java.util.Scanner;
class SimpleInterest {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter Principal: ");
    double p = sc.nextDouble();
    System.out.print("Enter Rate: ");
    double r = sc.nextDouble();
    System.out.print("Enter Time: ");
    double t = sc.nextDouble();
    double si = (p * r * t) / 100;
    System.out.println("Simple Interest = " + si);
  }
}
```

8. Multidimensional Arrays with Example

Found on (Page 106–108):

- In Java, **multidimensional arrays are arrays of arrays**.
- Syntax: int[][] arr = new int[3][3];
- Example below creates a 2D array and prints values:

```
class TwoDArray {
  public static void main(String[] args) {
    int[][] arr = new int[2][3];
    int k = 1;
    for(int i=0;i<2;i++){
```

```
        for(int j=0;j<3;j++){
          arr[i][j] = k++;
          System.out.print(arr[i][j] + " ");
        }
        System.out.println();
      }
    }
}
```