OOP

→ It utilizes concept of classes & objects. Its aim is to implement polymorphism, inheritance, encapsulation etc

→ Why is OOP used?:

1) To make development & maintenance of projects easier.

2) To hide & secure data

3) Provides solution to real-world problems.

① Package :

→ Grouping of related classes & interfaces into a single unit

→ Syntax: package packagename;

$\underset{\text{Keyword}}{\downarrow}$

→ Compilation:

```
javac -d. PackageExample.java

java package1. PackageExample.java
```

Date

Page no.

② Access Modifiers :

1) Private - data members & methods inside this aren't accessible from outside class.

2) Default - Only accessible ~~to~~ inside same package

3) Protected - Allows access inside same package & outside package inheritance.

4) Public - Can be accessed anywhere in program.

③ Class :

→ Entity that serves as basis for definition of new data type.

→ Syntax:

```
Modifier class className {

}
```

④ Object :

→ Used to access members of class.

→ Syntax:

```
Dog obj = new Dog();
```

(5) Static :

→ This keyword enables memory allocation & usage of variable or method only once.

→ Syntax:
```
static int a;
static void main();
```

(6) Final :

→ Once assigned to variable, we can't change its value.

→ Syntax:
```
final int a = 10;
      a = 100;
System.out.print(a);  → compilation error
```

(7) Constructors :

→ It's a member function of class & has same name as class.

→ Syntax:
```
public class Test {

        Test () {

    }
}
```

→ Default & parameterized are 2 types.

(8) This :

→ Used to refer instance variables of class
→ " " invoke constructor " "
→ " " " methods " "
→ can be passed as parameter in method call
→ " " " " " " constructor " .
→ " " used to return instance of class from method

(9) Super :

→ Used to refer to instance variable of parent class .
→ " " call methods of parent class .
→ " " " constructor " " " .

(10) Encapsulation :

→ It is process of grouping data members & corresponding methods into single unit .

→ It enables data hiding & secures data.

(11) Inheritance :

→ It enables code reusability
→ It also helps to achieve polymorphism .

Date

Page no.

→ Types :

1) Single-inheritence - one class extends to the other class

Syntax:
```
class Superclass{
}

class Subclass extends Superclass {
}
```

2) Multilevel inheritence - When class inherits from derived class & derived ~~cst~~ class becomes base class of new class.

Syntax:
```
class A {
}

class B extends ~~class~~ A {
}

class C extends B {
}
```

3) Hierarchical inheritance - one class serves as base class for more than one derived class.

Syntax:
```
class A {                    class C extends A {
}                            }
class B extends{
}
```

(12) Polymorphism :

→ Ability to perform single action in multiple ways.

(13) Interface :

→ Methods declared here are abstract i.e. they don't have body
→ It helps to achieve multiple inheritance

Syntax :
```
interface Test {

}
```

→ Multiple inheritance :-

```
interface Test 1 {

}

interface Test 2 {

}

public ~~class~~ class Main implements Test1, Test 2 {

}

public static void main (String[] args) {
    Main obj = new Main ();
}
```