**Day 6:**

sc.nextLine() hold the enter key.

String name = Sc.nextLine(); more than one word.

**Oops relationship**

1. Manager/Developer is a Employee
2. has a

Inside one class we are creating another class object is known as has a relationship.

class Employee {

id,name,salary

Address add = new Address();

Scanner

readEmp()

add.readAdd();

disEmp();

}

class Manager extends Employee {

numberOfEmp

readMgr()

disMgr()

}

class Developer extends Employee{

projectName;

readDev()

disDev();

}

class ProjectManager extends Manager{

clientInfo;

readPmgr()

disPmgr();

}

class Address {

city

state

Scanner

readAdd()

disAdd();

}

Has a relationship divided into 3 types.

1. Association
2. Aggregation
3. Composition

class A{

B obj1 = new B(); 1 or many

}

class B {

A obj2 = new A(); 1 or many

}

Either any one of the side we need to create another class object to achieve has a relationship.

class Employee {

Address padd = new Address();

Address oadd = new Address();

}

class Address {

}

This is a type of association but it is known as weak association. Weak association is known as aggregation.

Class Student{

StudentHistory sh = new StudentHistory();

}

Class StudentHistory {

}

This is type of association it is strong association. Strong association is known composition.

Polymorphism : One name many forms or many implementations.

2 types

1. Compile time or static binding or early binding

Method overloading:

Method have same name but different parameter list (number of parameter list or type of parameter list must be different).

1. Run time or late binding or dynamic binding

Method overriding:

Method have same name and same method signature (number of parameter list, type of parameter list and return type must same) only coding is different.

To achieve method overriding we need inheritance concept.

Annotation : annotation is meta-data. Data about data. Java provided lot of pre defined annotation. All annotation start with pre-fix @ followed by annotation. Few annotation we an use on class level or method level or property level.

**@Override** : this annotation we can use on method level. If method is overriding of super class we doesn’t get any error. Else we will the error.

Abstract keyword :

1. abstract keyword we can use with method and class but not with variable.
2. abstract method : method without body or incomplete method or without curly braces is known as abstract method.

abstract returntype methodName(parameterList);

1. if class contains one or many abstract method that class we need to declare as abstract class.

abstract class className {

}

1. whichever class extends abstract class that class must be provide the body for all abstract method mandatory.
2. Abstract class can contains normal as well as abstract method. It can contains one or many or zero abstract method.