Agenda

```
JVM Architecture
Final Field
Association
    Aggegration
    Composition
Inheritance
Types of inheritance
Access Modifiers
Upcasting
Dynamic Method Dispatch
```

JVM Architecture

```
1. ClassLoader
    a. Bootstrap CL
    b. Extension CL
    c. Application CL

2. Runtime Data Areas
    a. Method Area
    b. Java Heap
    c. Java Stack
    d. PC registers
    e. Nativ Method Stack

3. Execution Engine
    a. Interpreter
    b. JIT Compiler
    c. Garbage Collector
```

Final Field (Demo01)

```
- If you dont want to change the value of the fields once initiaized then
make such fields as final
- If the value of the field is fixed and does not change in any no of
objects of that class then make such fields as final as well as static
    eg- final static double pi = 3.14
```

Association

```
- If has-a relationship exists between two entities we use assocation
- If we cretae the reference of dependecny class as a field inside
```

```
dependent class then we call it as association.

- eg - Person has-a dob

1. Aggegration
    - If the entities are loosely coupled
    - In this the dependency class object is cretaed oustside the dependent
class and then passed to it.
2. Composition
    - If the entities are tightly coupled
    - In this the dependency class object is cretaed within the dependent
class.
```

Inheritance

```
- If is-a relationship exists between two entities then we use inheritance
- eg - Employee is-a Person
     - Student is-a Person
     - Circle is-a Shape
     - Rectangle is-a Shape

- Person or Shape class is called as Super(Parent) class
- Employee,Student,Cirle,Rectangle are called as sub(Child) class.
```

Super (Demo02 & Demo03)

```
- It is a statement that is used to decide which constructor you want to
call of the Super(parent) class
from your sub(Child) class constructor
- Super statement should be the first statement inside your sub(Child)
class ctor
- You can call the super class method from sub class methods using super.
```

Access Modifiers (Demo04)

```
- private members are accessiable only within the class and not outside the
class
- public members are accessiable in any classes in any packages
- protected and default are accessiable with the package and not outside
the package
- protected members are also accessiable inside the subclass irrespective
of package
```

upcasting & Dynamic Method Dispatch(Demo05)

- If you keep object of subclass into reference of super class then we call it as upcasting
- the reference of super class can call the overriden methods of subclass at runtime depending on which sub class instance it points to, this is called as dynamic method dispatch.