Agenda

```
Inheritance
Types of Inheritance
Method Overriding
Dynamic Method Dispatch
Upcasting
Downcasting
instanceof
final method
final class
Object class
toString
Abstract method
Abstract class
```

Inheritance

```
- is-a relationship
- Parent -> Super
- Child -> Sub
```

Super (Demo01)

```
- super is a keyword that can be used to call the super class Ctor as well
as super class methods
```

Method Overriding (Demo 02)

```
- If we keep the name and signature of method in sub class same as that of
super class then it is called as method overriding
- Why to do method overriding?
    1. To avoid confunsions at the time of method call on sub class
instance.
    2. To make the generalized methods specialized.
```

Types of (class)Inheritance

```
1. Single Inheritance
2. Hirerachical
3. Multilevel
```

```
- java does not support multiple class inheritance
- java does not support multiple implementation inheritance
```

upcasting

```
keeping the objects of subclass into super class reference is called as
upcasting
```

Dynamic Method Dispatch

```
- The reference of superclass will always call the methods of the object to
which it is pointing.
- This dynamic behaviour of calling the methods at run time is called as
Dynamic Method Dispatch
```

Downcasting (Demo04)

```
- Converting the superclass reference to the sub class reference is called
as downcasting.
- At the time of downcasting the super class reference should point at the
same object of the sub class.
- If the Super class reference is pointing at some differnt object and
downcasting is done then we get the exception called as ClassCastException
```

instanceof (Demo05)

```
- instanceof operator is used to check the reference is holding which type
of object.
```

Final Method (Demo06)

```
- If the super class method is complete and we dont want to allow the sub
classes to override it then make such super class methods as final
```

Final Class (Demo06)

```
- If the super class implementation is 100% complete and you dont want the
super class to be inherited into any subclasses then make the class as
final
```

Object class

```
- It is superclass of all the classes in java
- Object class is inside java.lang package
- Object class has total 11 Methods
- For eg few methods are listed below

    - boolean equals(Object obj)
    - String toString()
```

toString (Demo07)

```
- Object class has a method toString() which is responsible to return the
String representation of state of the object.
- If user defined class do not override toString method of object class
then object class toString method gets called with represents the state of
object in a way as below.
    - com.sunbeaminfo.entities.Person@6ff3c5b5
    - FullyQualifiedClassName@HashCode
```

Abstarct class & method (Demo 09)

```
- If user do not know the implementation of the method and do not want to
provide the method body then such methods can be made as abstract
- abstact methods should be kept inside abstact classes.
- If a class is abstract you cannot create object of such class.
- You can create only reference of the abstact class
- Abstact class is used to keep the method design same acress all related
types of classes.
```

equals Method (Demo 10)

```
- If you want to check the equality between user defined class objects then
you have to override the object class equals method
- The logic on which the equlity needs to be checked can be implemented as
per the user requirement
- the equals method return boolean value so we should return true or false
depending on the conditions.
- As equals method catches the user objects inside parameter of type Object
we need to do the downcasting.
- Before downcasting check conditions for null and instance of.
```

Lab Work

```
- Inheritance Demo
- Abstraction Demo
```