Day 10

- Deprecated means Type/Field/method is available to use in Java but not recommended to use because better alternative is available.
- System Date using java.util.Date class

```
Date date = new Date();
int day = date.getDate();
int month = date.getMonth() + 1;
int year = date.getYear() + 1900;
```

- System Date using java.util.Calendar class

```
Calendar c = Calendar.getInstance();
int day = c.get( Calendar.DAY_OF_MONTH);
int month = c.get(Calendar.MONTH) + 1;
int year = c.get(Calendar.YEAR);
```

- Using java.time.LocalDate

```
LocalDate ldt = LocalDate.now();
int day = ldt.getDayOfMonth();
int month = ldt.getMonthValue();
int year = ldt.getYear();
```

- Has-a, is-a, use-a and creates-a are class hierarchies.

Association

- Consider Examples:
    1. Room has-a Chair
    2. Room has-a Wall
    3. Car has-a engine
    4. Car has-a audio system
    5. Human has-a heart
    6. Department has-a faculty.
- If "has-a" realationship is exist between two types(class) then we should use association.
- How will you relate Car and Engine?
    1. Car has-a engine.
    2. Engine is a part of Car.
- Has-a hierarchy is also called as part-of hierarchy.

- Engine is part of car in other words engine instance(BS6) is a part of car instance(Maruti Suzuki - Alto).
- From object oriented point of view, if instance is a part of instance means it must get space inside another instance.

```
class Engine{
}
class Car{
    private Engine e;//Association
    public Car(){
        this.e = new Engine( );
    }
}
//Engine Instance   :   Dependancy Instance
//Car Instance      :   Dependant Instance
class Program{
    public static void main(String[] args) {
        Car c = new Car( );
    }
}
```

- If we declare instance(object) of a class as field inside another class then it is called association.
- In Java, association do not represent physical containment. In Java instance can be part of another instance using reference variable.
- Association => Instance is outside instance.
- How will you relate Person, date and address?
- Java Archive(jar) is a library file of Java that we can use to create reusable component.
- ".jar" files contains:
    1. META-INF - Manifest file
    2. Resources(images/audio files/fonts)
    3. Packages
- Create "associationlib.jar" file
- Project : AssociationLib

```
package org.sunbeam.dac.lib;
class Date{
    private int day, month, year;
}
class Address{
    private String cityName, stateName, pincode;
}
class Person{
    private String name;         //Association
    private Date birthDate;     //Association
    private Address address;    //Association
}
```

- Project : AssociationTest

1. include "associationlib.jar" into classpath/runtime classpath / build path.
2. import org.sunbeam.dac.lib package.

```
class Program{
    public static void main(String[] args) {
        Date date = new Date( );
        Address address = new Address( );
        Person person = new Person( );
    }
}
```

- Association has 2 special forms:
    1. Composition
    2. Aggregation

Composition

- Consider Example of Human and Heart
    - Human has-a heart. //Association

```
class Heart{
}
class Human{
    private Heart h = new Heart( ); //Association => Composition
}
//Heart Instance    :   Dependancy Instance
//Human Instance    :   Dependant Instance
```

- In case of association, if dependancy instance can not exist w/o Dependant instance then it is called composition.
- Composition represents tight coupling.

Aggregation

- Consider Example of Faculty and Department
    - Department has-a faculty.

```
class Faculty{
}
class Department{
    private Faculty f = new Faculty( ); //Association => Aggregation
}
//Faculty Instance    :   Dependancy Instance
//Department Instance    :   Dependant Instance
```

- In case of association, if dependancy instance can exist without Dependant instance then it is called as aggregation.
- Aggregation represents loose coupling.
- Consider Employee has-a joindate

```
class Date{
}
class Employee{
    private Date joinDate = new Date( ); //Association
}
```

Inheritance

- Consider Example
  1. Car is a Vehicle
  2. Book is a Product
  3. Manager is a Employee
  4. SavingAccount is a Account
- If "is-a" relationship is exist between two types then we should use inheritance.
- If we want to create child class / extend class then we should use extends keyword.

```
class Person{   //Parent class / Super class
}
class Employee extends Person{  //Child class / Sub class
}
```

- In Java, parent class is called super class. Child class is called sub class.
- If we create instance of super class then all the non static fields declared in super class get space inside it.
- If we create instance of sub class then all the non static fields declared in super class & sub class get space inside it.
- If we create instance of sub class then all(private/default/protected/public) the non static fields declared in super class get space inside it. In other words, non static fields of super class inherit into sub class.
- Using sub class name, we can use static field declared in super class. In other words, static fields of super class inherit into sub class.
- Fields of sub class do not inherit into super class. All the fields of super class inherit into sub class but only non static field get space inside instance sub class.
- We can call/invoke, non static method of super class on instance of sub class. In other words, non static methods of super class inherit into sub class.

- We can call, static method of super class on sub class. In other words, static methods of super class inherit into sub class.
- Points to remember:
  1. All the non static members(fields + methods) of super class inherit into sub class.
  2. All the static members(fields + methods) of super class inherit into sub class.
  3. Nested type of super class inherit into sub class.
  4. Constructor do not inherit into sub class.
- Every super class is abstraction for the sub class.
- If we create instance of super class then only super class's constructor gets called.
- If we create instance of sub class then first super class and then sub class constructor gets called.
- From any constructor of sub class, by default, super class's parameterless constructor gets call.
- Using super statement, we can call any constructor of super class from constructor of sub class.
- Super statement must be first statement inside constructor body.
- Inheritance is a process of acquiring/accessing/getting properties(fields) and behavior(methods) of super class inside sub class.
- Inheritance is also called as Generalization.
- According to client's requirement, if implementation of exisiting class is partialy complete/logically incomplete then we should extend that class i.e we should use inheritance.
- According to client's requirement, if implementation of super class method is incomplete then we should redefine that method inside sub class.
- If name of members of super class and sub class are same and if we try to access these members using instance of sub class then preference is given to the sub class members.
- If name of members of super class and sub class are same and if we try to access these members using instance of sub class then sub class members hides implementation of super class members. This process is called shadowing.
- Using super keyword, we can use any member of super class inside method of sub class.