Day 5

Enum

```
- If we want to improve readability of source code then we       should
use enum.
- Enum is reference type.
- If we want to give name to the constant/literal then we        should
use enum.
- If we want to define enum then we should use enum keyword
```

```
enum Shape  //Enum Name
{
    LINE,RECT,OVAL  //Enum Constant
    //LINE=0,RECT=1,OVAL=2  //0,1,2 -> Ordinal
}
```

```
- java.lang.Enum is considered as super class for all the enums in java.
It is abstract class.
- Methods of java.lang.Object class
    1. public String toString( );
    2. public boolean equals( Object obj );
    3. public native int hashcode( );
    4. protected native Object clone()throws CNSE
    5. protected void finalize( )throws Throwable
    6. public final Class<?> getClass( );
    7. public final void wait( )throws InterruptedException
    8. public final native void wait( long timeout )throws IE
    9. public final void wait(long timeout,int nanos)throws IE
    10. public final native void notify();
    10. public final native void notifyAll();
- Methods of java.lang.Enum class
    1. public final Class<E> getDeclaraingClass( );
    2. public final String name( );
    3. public final int Ordinal( );
    4. public static T valuof( Class<T> enumType, String name);
- Sole constructor : A constructor which is designed to call from sub
class constructor only.
- Java Compiler generates .class file per interface, class and enum.
- After compilation java compiler convert enum as follows
```

```
final class ShapeType extends Enum<ShapeType>
{
  public static final ShapeType LINE;

  public static final ShapeType RECT;

  public static final ShapeType OVAL;

  public static ShapeType[] values();

  public static ShapeType valueOf(String name );
}
```

```
 - In java, enum is implicitly considered as final class hence we can not
extend enum.
- "values()" and "valueOf()" are static methods of enum added at compile
time.
- In C/C++, If we want to assign name to the literal then we should use
assignment operator.
- In C, using enum, we can assign name to the integer literals only( int,
char etc ).
```

```
enum Day
{
    SUN=1,MON=2,TUES=3,WED=4
}
```

- In java, If we want to assign name to the literal then we should use parethesis operator[()].
  - In java, using enum, we can assign name to single as well as group of literals of any type.
  - e.g SUN(1); MON("MonDay "); TUES(3, "TuesDay");
  - If we want to assign name to the literals then we should define constructor inside enum.
  - Enum constant declaration statement must be first statement inside enum.
  - In java, Inside enum we can
    1. Declare Fields
    2. Define constructor, ,methods
    3. Override methods of java.lang.Object class.

Major Pillars of OOPS

```
1. Abstraction
2. Encapsulation
3. Modularity
4. Hieraracy
```

```
- Here word "major" means, language without any one of the above feature
will not be object oriented.
```

Minor Pillars of OOPS

```
1. Typing / Polymorphism
2. Concurrency
3. Persistence
- Here word "monor" means, above features are useful but not essential to
classify language object oriented.
```

Abstraction

```
- It is major pillar of oops
- Abstraction defines outer behavior of object/instance
- Process of getting essential things from system is called abstraction.
- Abstraction focuses on outer behavior of some object relative to the
perspective of viewer
- Main purpose of abstraction is to achive simplicity
- Following code describe abstraction programatically:
    Complex c1 = new Complex();
    c1.acceptRecord();
    c1.printRecord();
```

Encapsulation

```
- It is major pillar of oops
- Encapsulation defines internal behavior of object/instance
- To achive abstraction, we need provide some implementation.    It is
called encapsulation.
- Binding of data and code together is called encapsulation
- Main purpose of encapsulation is:
    1. To achive abstraction
    2. To achive data hiding
- Following code describe encapsulation programatically:
    class Complex
    {   //Data
        private int real;
        private int imag;
        //Code
        public void acceptRecord( )
        {   }
        public void printRecord( )
        {   }
    }
- Absraction ans encapsulation are complementry concepts. Abstraction
focuses on outer bahavior whereas encapsulation focuses on internal
behavior.
```

Modularity

```
- It is major pillar of oops
- It is process of developing complex system using small modules/parts
- Main purpose of modularity is to minimize module dependancy
- In java, we can achive modularity with the help of libraries(.jar, .war,
.ear )
```

Hierarchy

```
- It is major pillar of oops
- Level/Order/Ranking of abstraction is called hierarchy.
- Main purpose of hierarchy is to achive reusability.
- Reusability help us :
    1. To reduce development time and
    2. To reduce development cost
- Types of hierarchies
    1. "Has-a" / "Part-of"  ->  Association / Containment
    2. "Is-a" / "Kind-of"   ->  Inheritance / Generalization
    3. "Use-a"              ->  Dependancy
    4. "Creates-a"          ->  Instantiation
```

Association

```
- If "has-a" relationship exist between two types then we should use
association.
- Let us consider example of car:
    - Car has-a engine in other word engine is part of car
    class Engine
    {   }
    class Car
    {
        Engine e = new Engine();   //Association
    }
    Car c = new Car();
- If object is part of / component of another object then it is called
association.
- In above Code:
    Dependant Object is : Car Object
    Dependancy Object is : Engine Object
- In case of association, we should declare instance of reference type as
a field inside another class.
- Association has two special forms:
    1. Aggregation
    2. Composition
```

Compostion

```
- It is special form of association
- Example : Human has-a heart
    class Heart
    {   }
    class Human
    {
        Heart h = new Heart();  //Association-->Composition
    }
    Human h = new Human( );
    Dependant Object    : Human Object
    Dependancy Object   : Heart Object
- In case of association, if dependancy object do not exist without
dependant object then it is called Composition.
- Composition represents tight coupling
```

Aggregation

```
- It is special form of association
- Example : Room has-a chair
    class Chair
    {   }
    class Room
    {
        Chair ch = new Chair();
    }
    Room r = new Room();
    - Dependant Object    : Room Object
    - Dependancy Object   : Chair Object
- In case of association, if dependancy object exist without dependant
object then it is called aggregation.
- Aggrgation represents loose coupling
```

Java Archive(jar)

```
- AssociationLib( associationlib.jar)
    - Date[ day, month, year, ctor, getter,setter, toString ]
    - Address[ city,state,pin, ctor, getter,setter, toString ]
    - Person[ name, dob, addrs, ctor, getter,setter, toString ]
- AssociationTest( refer associationlib.jar in this project )
    - Program
```

```
class Date
{   }
class Address
{   }
class Person
{
    //Association
    private String name = new String();
    private Date birthDate = new Date();
    private Address currAddress = new Address();
}
class Program
{
    public static void main(String[] args)
    {
        //TODO : Test functionality of Date, Address and Person
    }
}
```

System Date Time using Calendar

```
- Calendar is abstract class declared in java.util package
```

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

System Date Time using Date

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

System Date Time using LocalDate

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

Steps to create jar file

```
1. Create java project and define types inside it.
2. Right Click on java project --> Export --> Java--> JAR file
    --> Next--> Select resource(s) --> Select location for jar
    file --> Finish
```

Steps to use jar file

```
1. Create java project and add jar file in classpath/runtime classpath/
buildpath
2. Right Click on project --> Build Path --> Configure Build Path --> Java
Build Path --> Libraries --> Add External Jars --> Choose jar file and
click on open button. --> Click on Appy --> Apply and Close.
```

Plain Old Java Object(POJO)

```
- A class which do not extend class or implement any interface and follows
following rule is called POJO class
    1. It must be packaged public class
    2. It must contain default(parameterless) constructor
    3. Fields must be private
    4. For every private field getter/setter methods must be exist inside
class
    5. Should not contain business logic method
    6. can contain toString, equals and hashcode method.
- It is mainly used for object reltational mapping( ORM )
    class<--->Database Table
- POJO class is also called Data Transfer Object( DTO ) / Value Object( VO
) / Business Object( BO ) / Entity
```

Inheritance

```
- Example:
    1. Car "is-a" vehicle
    2. Manager "is-a" Employee
    3. Laptop "is-a" mobile
    4. Book "is-a" product
- If "is-a" relationship exist between two types then we should use
inheritance.
- Inheritance is also called as generalization.
- Field is also called as attribute, property, data member
- Method is also called as Operation, behavior, message, member function.
- In java Parent class is called Super class and child class is called sub
class.
- Syntax:
```

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

```
- During inheritance, properties(field) and behavior( method) of super
class inherit into sub class.
- During inheritance all fields( static + non static ) of super inherit
into sub class but only non static fields get space inside instance.
- Except constructor, all methods( static + non static ) of super class
inherit into sub class.
- During inheritance nested types inherit into sub class.
- using sub class instance, we can call/access method of super class i.e
non static method inherit into sub class.
- Using sub class name, we can access static method of super class i.e
static method inherit into sub class.
- If we create instance of sub class then non static fields declared in
super class get space inside it. In other words, non static fields inherit
into sub class.
- All the private members of super class inherit into sub class.
- If we create instance of sub class then First super class and then sub
class's constructor gets called.
- From sub class constructor, by default, super class's parameterless
constructor gets called.
- If we want to call, constructor of super class from constructor of sub
class then we should use super statement.
- Super statement must be first statement inside constructor body.
```