Dependancy Injection
==========
   The process of injecting dependant object into target object is called as "Dependancy injection".

We can achieve dependancy injection in 2 ways
      a. Constructor dependancy injection
      b. Setter dependancy injection

a. Constructor dependancy Injection
      Injecting dependant object into target object through a constructor is called as "Constructor Dependancy Injection".

b. Setter dependancy Injection
      Injecting dependant object into target object through a setter is called as "Setter Dependancy Injection".

eg#1
```
class Engine { //Dependant Object

}
class Car{//Target Object
            //HAS-A relationship
      Engine engine ; //instance variable
}
```

eg#2
```
class Address{//Dependant Object

}

class Student {//Target Object
      //HAS-A  relationship
      Address address; //instance variable
}
```

eg#3
```
class Account{//Dependant Object

}
class Employee {//Target Object
      //HAS-A relationship
      Account account;//instance variable
}
```

Relationships in JAVA
==========
As part of Java application development,we have to use entities(class) as per the application requirements.
In Java application development,if we want to provide optimizations over memory utilization,code Reusability, Execution Time,Sharability
then we have to define relationships between entities.

There are three types of relationships between entities.
1.Has-A relationship (extensively used in projects)
2.IS-A relationship (achieved using extends keyword)
3.USE-A relationship(not popular)

Q)What is the difference between HAS-A Relationship and IS-A relationship?
Ans:
Has-A relationship will define associations between entities(class) in Java applications,here associations between entities will
improve communication between entities and data navigation between entities.

IS-A Relationship is able to define inheritance between entity classes, it will improve "Code Reusability" in java applications.

Associations in JAVA
==========
There are four types of associations between entities
1.One-To-One Association(1:1)
2.One-To-Many Association (1:M)
3.Many-To-One Association (M:1)
4.Many-To-Many Association (M:M)

To achieve associations between entities,we have to declare either single reference or array of reference variables of an entity class in
another entity class.

```
class Address{
      ----
}
class Account{
      ----
}
class Employee{
      ----
      Address[] addr;//It will establish One-To-Many Association (1:M)
      Account account; // One-To-One Association(1:1)
}
```

1.One-To-One Association:
It is a relation between entities, where one instance of an entity should be mapped with exactly one instance of another entity.
      eg: Every employee should have exactly one Account.

eg: refer:: 02-One-One-Association-mapping

2.One-To-Many Association:
It is a relationship between entity classes, where one instance of an entity should be mapped with multiple instances of another entity.
      Example:Single department has multiple employees.

eg: refer:: 03-One-MANY-Association-mapping

Many-To-One Association:
==========
It is a relationship between entities,where multiple instances of an entity should be mapped with exactly one instance of another entity.
      EX:Multiple Student have joined with a single branch.

eg: refer:: 04-MANY-ONE-Association-mapping

Many-To-Many Associations
==========
It is a relationship between entities,Where multiple instances of an entity should be mapped with multiple instances of another entity.
      EX:Multiple Students Have Joined with Multiple Courses

eg: refer:: 05-MANY-MANY-Association-mapping