```
1.Aspect
2.Advice
3.PointCut
4.JoinPoint
5.Target
6.Weaving
7.Proxy

Annotations
===========
1.@Aspect
2.@Before,@After,@Around,@AfterReturning,@AfterThrowing
3.@PointCut

=>@PointCut("execution(""))

PointCut
========
 It is an Expression which selects Buisness class methods which needs advices.
 PointCut can never specify which Advice is going to be selected.

Pointcut Syntax
===============
 Specifier ReturnType package.ClassName.methodName(parameterType)
      Note: Symbols allowed in PointCut Expression : .(dot),*(star)

Examples

1. public int in.ineuron.dao.EmployeeDao.saveEmployee(Employee)
      saveEmployee() method having parameter Employee with return type 'int' of
type public defined inside a class EmployeeDao(in.ineuron.Dao)
      is selected to connect with Advice.

2. public int in.ineuron.dao.EmployeeDao.*()
       => Zero parameter
       => Any methodName/Method inside EmployeeDao
       => int return type

3. public * in.ineuron.dao.EmployeeDao.*(..)
      => Any no of parameter
      => Any methodName/Method inside EmployeeDao
     => Any return type

4. public * in.ineuron.dao.*.*()
      => All classes present in in.ineuron.dao packages and there methods which
accepts zero argument and method can have any return anytype.


B.Methods
=========
M#1 public int saveEmployee(Employee emp){}
M#2 public void deleteEmployee(Integer eid){}
M#3 public void updateEmployee(Employee emp){}
M#4 public Employee getEmployee(Integer eid){}

PointCut Expressions
====================
 a. public * *()
            [Zero params, any parameter any return type]
      No of methods matching => zero

 b. public void *(..)
         [Any no of params,and return type is void]
      No of methods matching(2) => M#2,M#3

 c. public * saveEmployee(..)
             [Any no of params, and return type]
            No of methods matching(1) => M#1

 d. public * *(Integer)
            [Only one param and can have any return type]
            No of methods matching(1) => M#4

Note:: * in the return type doesn't select a method with void return type.


Usage of AOP in realtime environment
====================================
interface EmployeeRepository extends CrudRepository<Employee,Long>{}

@Service
public class EmployeeServiceMgmtImpl implements IEmployeeService{
            @Transactional
            public void saveEmployee(Employee){}
}

Case2: If exception occurs in buisness method and if that exception information has
to be known by Advices then we need to use

EmployeeDao.java
================
@Component
public class EmployeeDao {
      public void saveEmployee() {
            System.out.println("Employee saved to database....");
            if (new Random().nextInt(15)<10) {
                  throw new RuntimeException("DUMMY EXCEPTION");
            }
      }
}

TransactionManagement.java
==========================
@Aspect
@Component
public class TransactionManagement {

      @Pointcut("execution(public * in.ineuron.dao.*.*(..))")
      public void p1() {}

      @AfterThrowing(value = "p1()",throwing = "exception")
      public void rollBackTx(Throwable exception) {
            System.out.println("Transaction rollbacked:: "+exception.getMessage());
      }
}
```

```
Case3: If there is a return type in buisness method and if that returned value has
to be known by Advices then we need to use

EmployeeDao.java
================
@Component
public class EmployeeDao {
      public String saveEmployee() {
            System.out.println("Employee saved to database....");
            return "Hello";
      }

}

TransactionManagement.java
==========================
@Aspect
@Component
public class TransactionManagement {

      @Pointcut("execution(public * in.ineuron.dao.*.*(..))")
      public void p1() {}

      @AfterReturning(value="p1()",returning = "obj")
      public void commitTx(Object obj) {
            System.out.println("Transaction commited..."+ obj);
      }

}
```