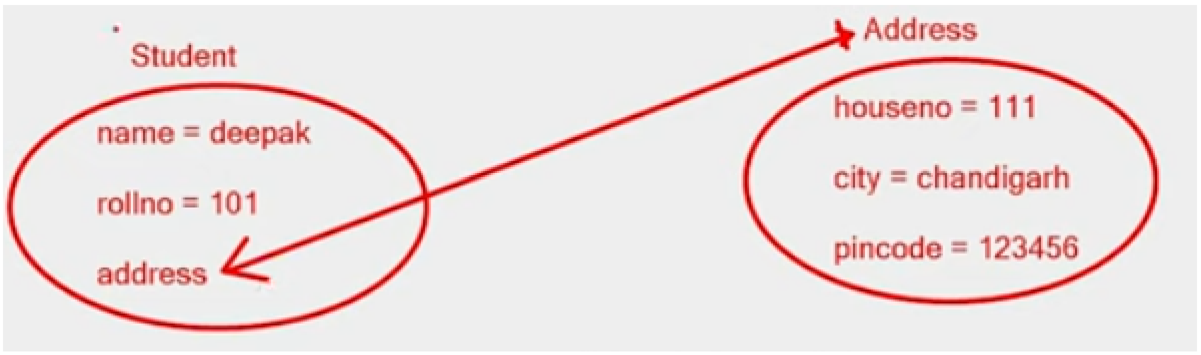
DEPENDENCY INJECTION:

1. Dependency Injection is a design pattern used in the Spring Framework to achieve Inversion of Control (IOC).
2. Its main task is to inject the dependencies, means injecting one object (a dependency) into another object.
3. We can achieve Dependency Injection by 2 ways :-
    - Setter Method DI
    - Constructor DI



```
public class Address
{
        private int houseno;
        private String city;
        private int pincode;
        public Address(int houseno, String city, int pincode)
        {
                System.out.println("Contructor from Address");
                this.houseno = houseno;
                this.city = city;
                this.pincode = pincode;
        }
        @Override
        public String toString() {

                return ("#"+houseno+" , "+city+" - "+pincode);
        }
}
```

```
public class Student
{
        private int rollno;
```

```java
	private String name;
	private Address address;
	public Student(int rollno, String name, Address address)
	{
		System.out.println("Contructor from Student");
		this.rollno = rollno;
		this.name = name;
		this.address = address;
	}
	public void display()
	{
		System.out.println("Roll no :"+rollno);
		System.out.println("Name : "+name);
		System.out.println("Address : "+address); //internally call toString() method in Address
	}
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="
     http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">
  <!-- bean definitions here -->
  <bean class="in.beans.Address" id="addr_id">
		<constructor-arg  value="2222"/>
		<constructor-arg  value="Piune"/>
		<constructor-arg  value="987654"/>
 </bean>
 <bean class="in.beans.Student" id="std_id">
		<constructor-arg value="111"/>
		<constructor-arg value="Saif"/>
		<constructor-arg ref="addr_id"/>
 </bean>
</beans>
```

```java
public class Main
{
	public static void main(String[] args) {
		String locString = "/in/res/applicationContext.xml";
		ApplicationContext context = new ClassPathXmlApplicationContext(locString);
		Student stdStudent = (Student)context.getBean("std_id");
		stdStudent.display();
	}
}
```

Setter Method DI :-
1. Dependencies are injected into a class through setter methods
2. Setter Method DI is more readable
3. Setter Method DI is more flexible

Constructor DI :-
1. Dependencies are injected into a class through constructor
2. Constructor DI is less readable
3. Constructor DI is less flexible

AUTOWIRING :

1. It is a feature of Spring Framework used to achieve **DI automatically**.
2. It can be achieved by :
    a. Annotation - @autowired and @Qualifier(check which obj has to inject)
    b. XML File - autowire attribute and mode - byName , byType , Constructor
        autowire-candidate=false(obj not involve in autowiring)

AUTO **WIRING**

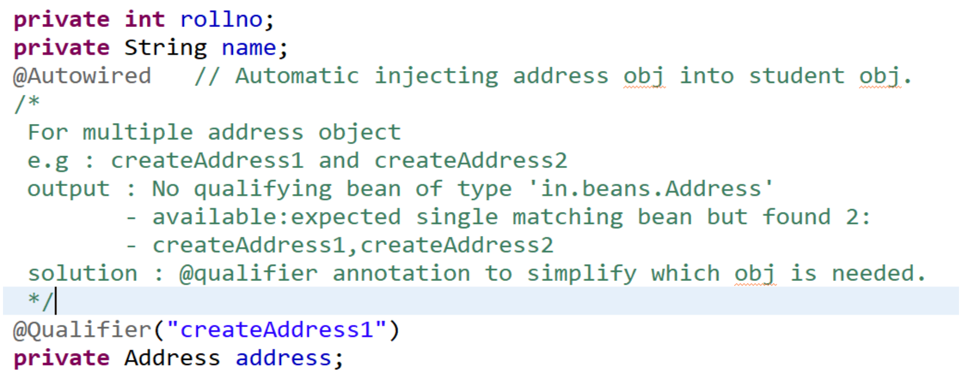
Automatically manage the Connection b/w objects Linking those objects to fulfill dependencies

Advantage :
- It requires less code

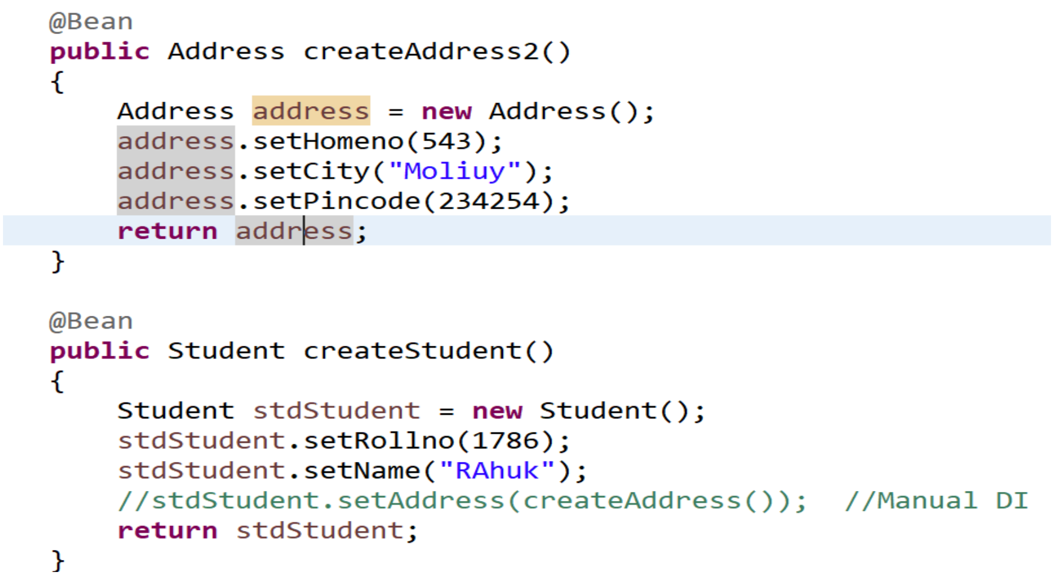
Disadvantage :
- It can be achieved only on non-primitive or user-defined data types (excluding String), not on primitive data types.

<u>Autowiring using Annotation :</u>

- Create POJO classes i.e student and address
- Create JavaConfig.class file use **@Configuration** to configure and **@Bean** to create POJO objects.
- Direct use **@Autowired** annotation with property to inject dependencies one to another.
- Create Main class and create a **applicationcontext** obj with annotation*** class.
- If there is multiple object and have to inject one object use **@Qualifier** annotation to simplify which object want to inject. E.g @Qualifier(“createobj”)



```
private int rollno;
private String name;
@Autowired   // Automatic injecting address obj into student obj.
/*
 For multiple address object
 e.g : createAddress1 and createAddress2
 output : No qualifying bean of type 'in.beans.Address'
        - available:expected single matching bean but found 2:
        - createAddress1,createAddress2
 solution : @qualifier annotation to simplify which obj is needed.
 */
@Qualifier("createAddress1")
private Address address;
```



```
@Bean
public Address createAddress2()
{
    Address address = new Address();
    address.setHomeno(543);
    address.setCity("Moliuy");
    address.setPincode(234254);
    return address;
}

@Bean
public Student createStudent()
{
    Student stdStudent = new Student();
    stdStudent.setRollno(1786);
    stdStudent.setName("RAhuk");
    //stdStudent.setAddress(createAddress());  //Manual DI
    return stdStudent;
}
```

Autowiring using XML file:

- Create POJO classes i.e student and address.
- Create XML config file.
- Create object using **<bean>** tag with attribute **class , id , autowire**
    - Autowire mode :-
        - byName - property , name , value
        - byType - property , name , value
        - constructor - constructor-arg , value , **index**
- Create Main class and create a **applicationcontext** obj with ClassPath*** class.
- If there is multiple object and have to inject one object use **autowire-candidate** attribute to simplify which object want to inject

1. **byName :**

```
<bean class="in.beans.Address" id="address1" autowire-candidate="false">
    <property name="homeno" value="453"/>
    <property name="city" value="Poand"/>
    <property name="pincode" value="132431"/>
</bean>

<bean class="in.beans.Student" id="stdid" autowire="byName">
    <property name="rollno" value="1011"/>
    <property name="name" value="Saiygtd"/>
    <!--property name="address" ref="addres"/ -->   <!-- Manual DI -->
</bean>
```

2. **byType :**

```
<bean class="in.beans.Address" id="addr2">
    <property name="homeno" value="453"/>
    <property name="city" value="Poand"/>
    <property name="pincode" value="132431"/>
</bean>

<bean class="in.beans.Student" id="stdid" autowire="byType">
    <property name="rollno" value="1011"/>
    <property name="name" value="Saiygtd"/>
    <!--property name="address" ref="addres"/ -->   <!-- Manual DI -->
</bean>
```

3. **Constructor :**

```
<bean class="in.beans.Address" id="addr2" autowire-candidate="false">
    <constructor-arg value="453"/>
    <constructor-arg value="Poand"/>
    <constructor-arg value="132431"/>
</bean>

<bean class="in.beans.Student" id="stdid" autowire="constructor">
    <constructor-arg value="1011" index="0"/>
    <constructor-arg value="Saiygtd" index="1"/>
    <!--property name="address" ref="addres"/ -->   <!-- Manual DI -->
</bean>
```