```
class Car {                                    class Engine {
Engine e;
                                               }
}
```

Car HAS_A Engine
Car Using Engine

Car is depending on Engine

Dependent	CAR
Dependency	Engine

```
class Student {                                POJO/Model/Domain/Data
        int m1,m2,m3;
}
```

here, Student is dependent
m1,m2,m3 are dependencies

Dependeny values to be injected to dependent object before using dependent object

if Student object created, we need to make sure that all dependeny values are injected into Student object

Using parameterized constructor while creating object , dependency values can be injected

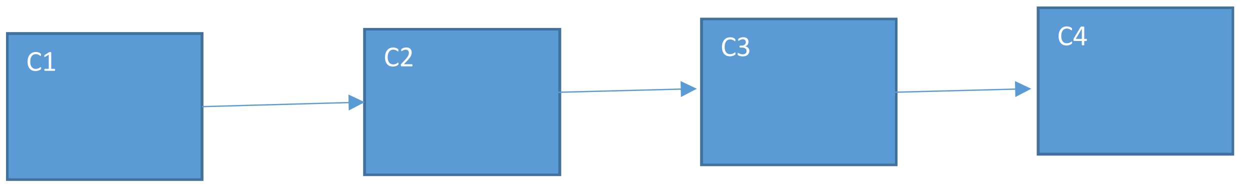
Student s1;

after creating object, inject dependency values into dependent using setters op1 Student s1 = new Student(56,76,78);

here, dependeny values are injected to dependent object while creating dependent object
thru parameterized constructor

op2 Student s2 = new Student();
s2.setM1(55);
s2.setM2(66);
s2.setM3(77);
here, depdent object created, after creating dependent object, injecting dependency values into dependent object'
using setters

Using op1 or op2, we need to make sure that dependent object has all dependency values injected before using object



create C4 object
create C3 object by injecting C4 object
create C2 objecet by injecting C3 object
create C1 object by injecting C2 object

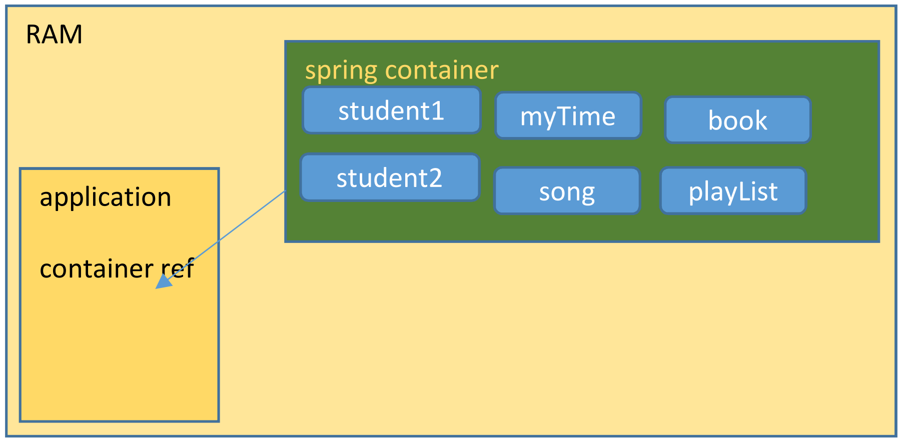
Programmer
- creates object
- injects all dependency values

Spring
- create object
- injects all dependency values

Spring creates dependent object by injecting all dependency values
such objects are called Beans

Spring container

Space in RAM where all beans loaded before using



class Student…

class MyTime..

class Author…

class Book…

class Song…

class PlayList…

student bean

mytime bean

author bean

Bean

Ready to use object by injecting all dependencies

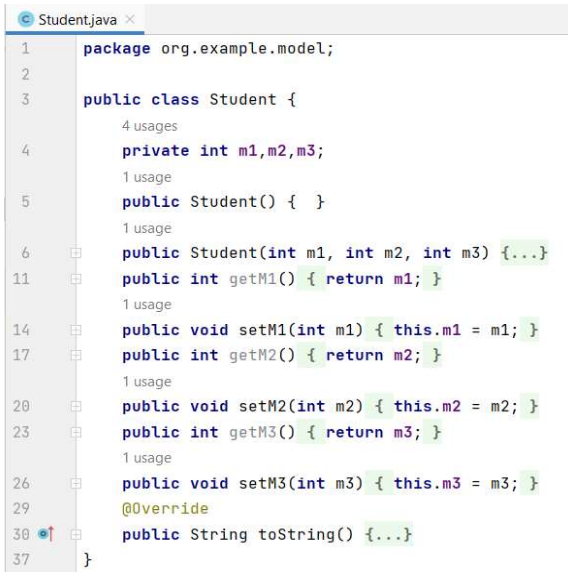
Spring creates object and injects all dependencies and loads this object into spring container, then this bean is ready to use

Factory

Factory method

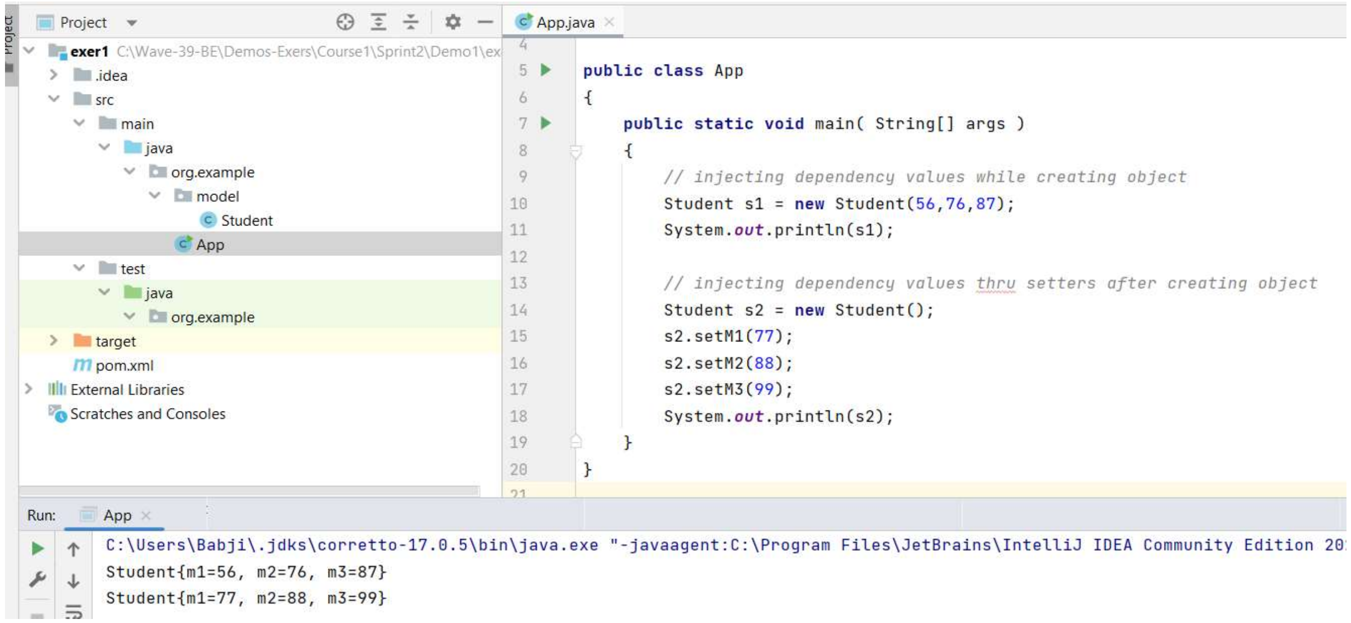
the method which provides object ready to use

Traditional way of creating object and using



Student.java

```java
package org.example.model;

public class Student {
    4 usages
    private int m1,m2,m3;
    1 usage
    public Student() {  }
    1 usage
    public Student(int m1, int m2, int m3) {...}
    public int getM1() { return m1; }
    1 usage
    public void setM1(int m1) { this.m1 = m1; }
    public int getM2() { return m2; }
    1 usage
    public void setM2(int m2) { this.m2 = m2; }
    public int getM3() { return m3; }
    1 usage
    public void setM3(int m3) { this.m3 = m3; }
    @Override
    public String toString() {...}
}
```

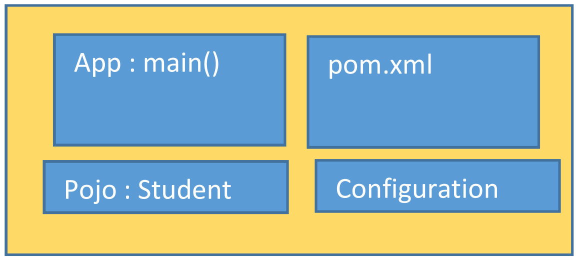


App.java

```java
public class App
{
    public static void main( String[] args )
    {
        // injecting dependency values while creating object
        Student s1 = new Student(56,76,87);
        System.out.println(s1);

        // injecting dependency values thru setters after creating object
        Student s2 = new Student();
        s2.setM1(77);
        s2.setM2(88);
        s2.setM3(99);
        System.out.println(s2);
    }
}
```

Run: App

```
C:\Users\Babji\.jdks\corretto-17.0.5\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 20
Student{m1=56, m2=76, m3=87}
Student{m1=77, m2=88, m3=99}
```



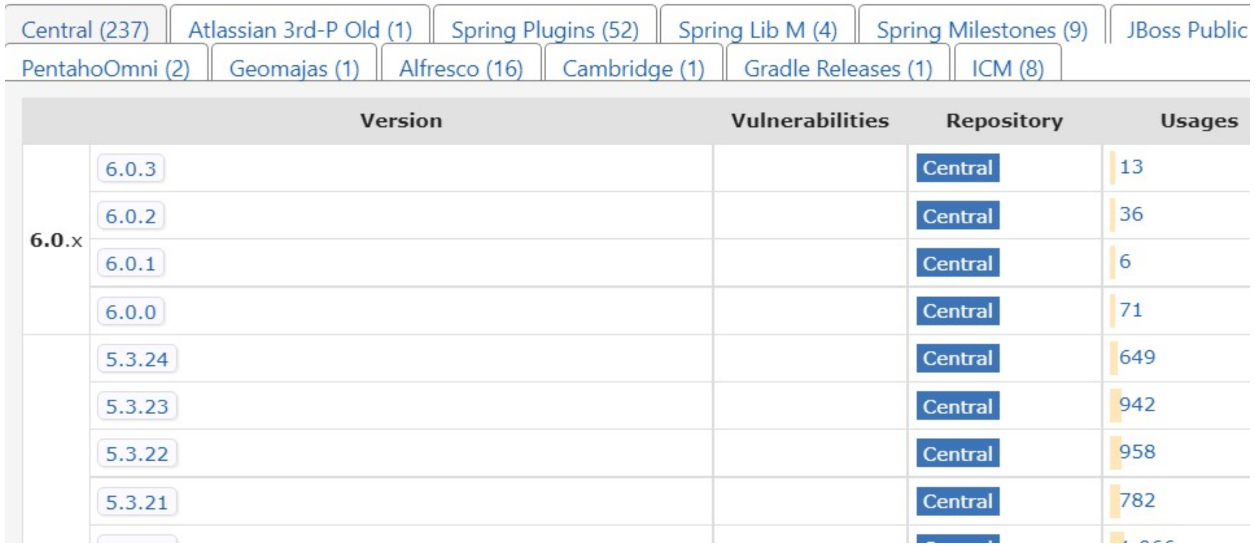
How to create beans using spring configuration, load beans into spring container, get beans from spring container and use

Demo1

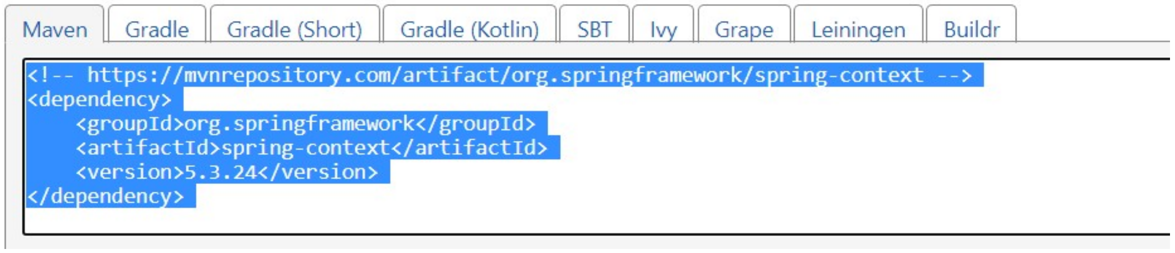
Step 1 add required dependency in pom

goto maven repository

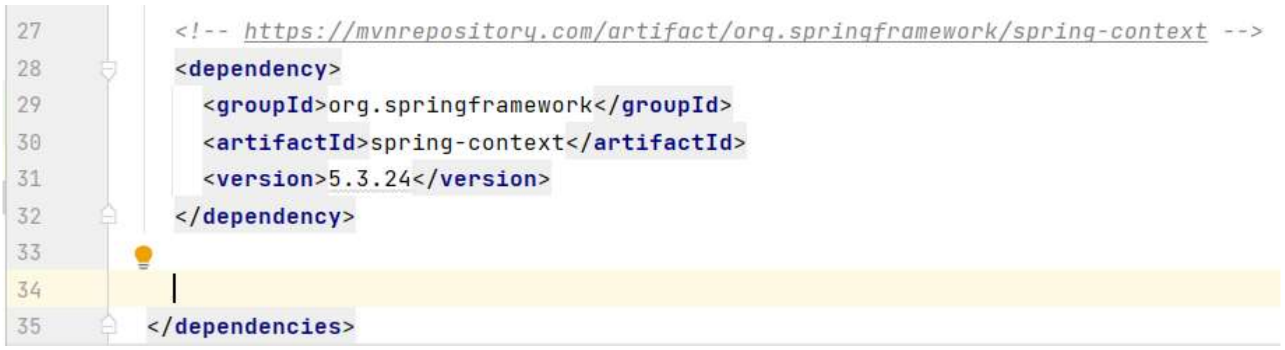
search for **spring context**



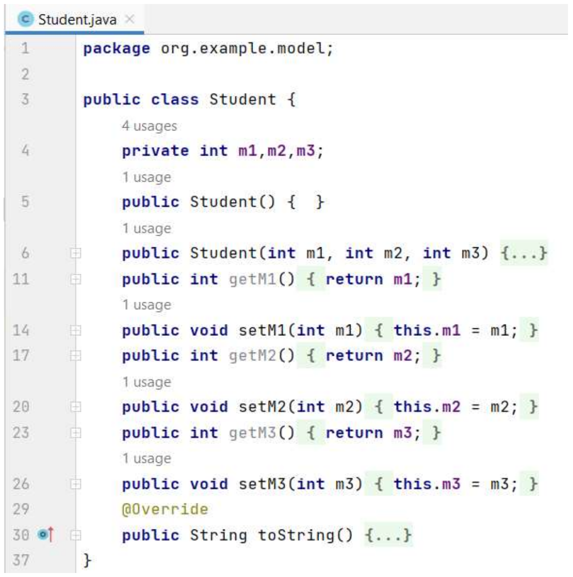
	Version	Vulnerabilities	Repository	Usages
6.0.x	6.0.3		Central	13
	6.0.2		Central	36
	6.0.1		Central	6
	6.0.0		Central	71
	5.3.24		Central	649
	5.3.23		Central	942
	5.3.22		Central	958
	5.3.21		Central	782



```
<!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.3.24</version>
</dependency>
```


```xml
<!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-context</artifactId>
  <version>5.3.24</version>
</dependency>


</dependencies>
```

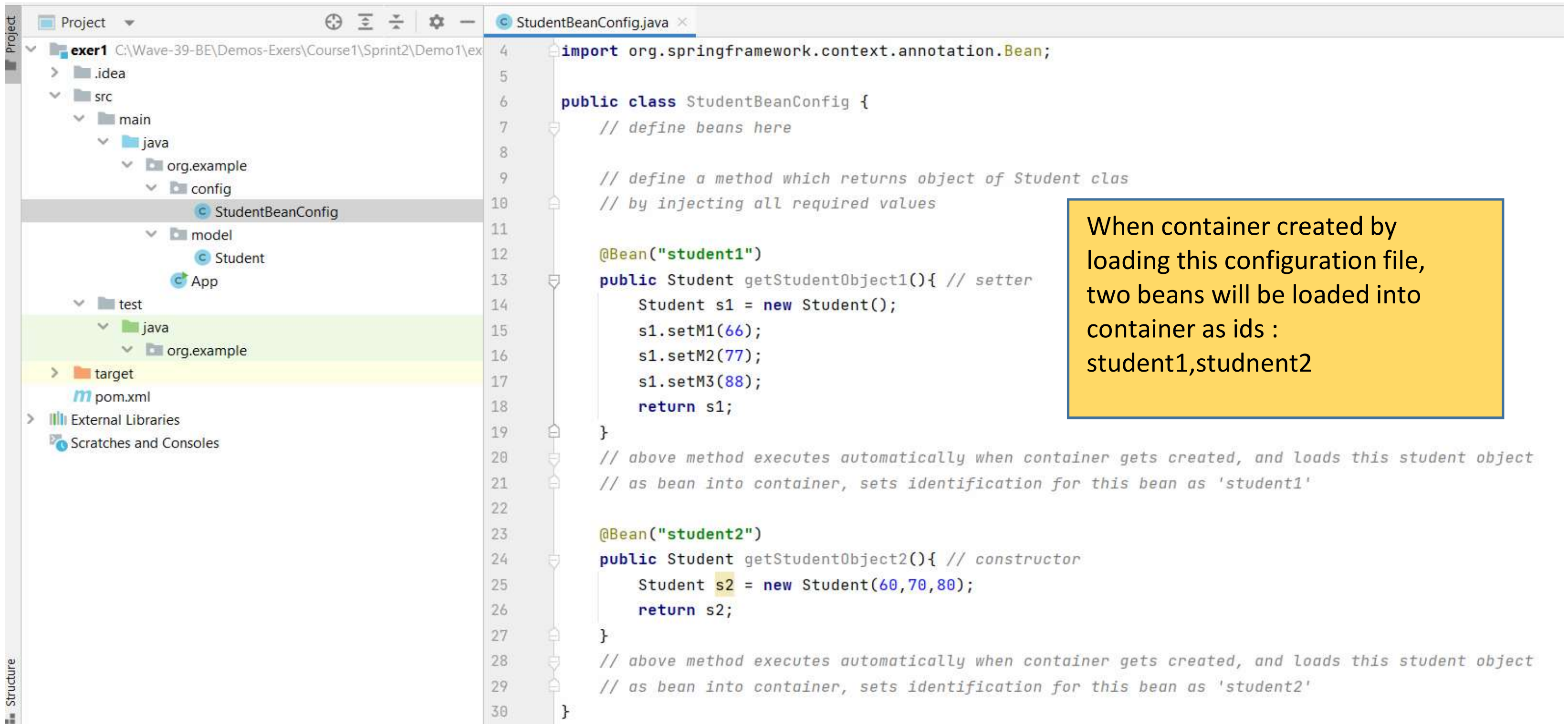
Step 2 Create Pojo class (domain/model/data)



```java
package org.example.model;

public class Student {
    4 usages
    private int m1,m2,m3;
    1 usage
    public Student() {  }
    1 usage
    public Student(int m1, int m2, int m3) {...}
    public int getM1() { return m1; }
    1 usage
    public void setM1(int m1) { this.m1 = m1; }
    public int getM2() { return m2; }
    1 usage
    public void setM2(int m2) { this.m2 = m2; }
    public int getM3() { return m3; }
    1 usage
    public void setM3(int m3) { this.m3 = m3; }
    @Override
    public String toString() {...}
}
```

with member variable
constructors
getters / setters
toString

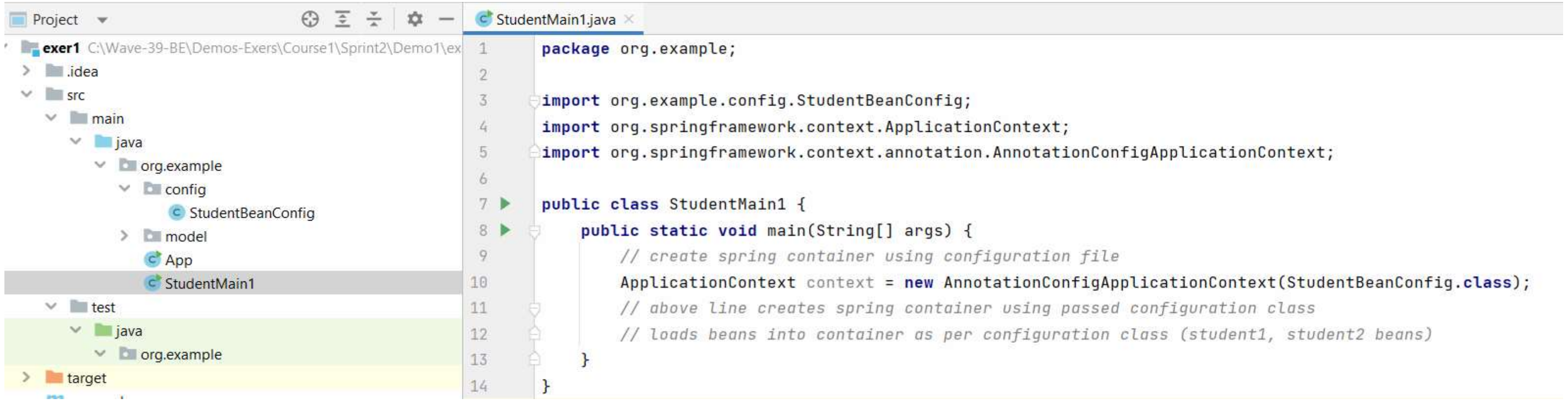
Step 3 Configure beans in configuration file



```java
import org.springframework.context.annotation.Bean;

public class StudentBeanConfig {
    // define beans here

    // define a method which returns object of Student clas
    // by injecting all required values

    @Bean("student1")
    public Student getStudentObject1(){ // setter
        Student s1 = new Student();
        s1.setM1(66);
        s1.setM2(77);
        s1.setM3(88);
        return s1;
    }
    // above method executes automatically when container gets created, and loads this student object
    // as bean into container, sets identification for this bean as 'student1'

    @Bean("student2")
    public Student getStudentObject2(){ // constructor
        Student s2 = new Student(60,70,80);
        return s2;
    }
    // above method executes automatically when container gets created, and loads this student object
    // as bean into container, sets identification for this bean as 'student2'
}
```

When container created by loading this configuration file, two beans will be loaded into container as ids : student1,studnent2

Step 4 in main()
create container using configuration file



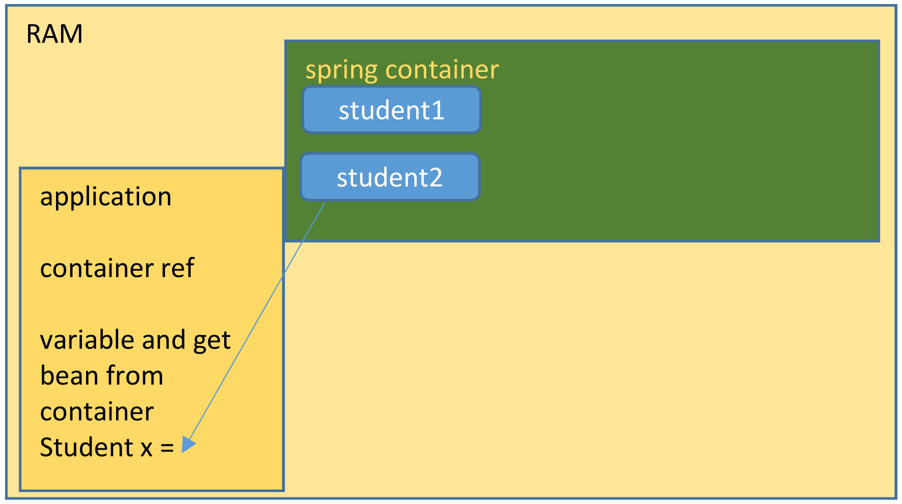
```java
package org.example;

import org.example.config.StudentBeanConfig;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class StudentMain1 {
    public static void main(String[] args) {
        // create spring container using configuration file
        ApplicationContext context = new AnnotationConfigApplicationContext(StudentBeanConfig.class);
        // above line creates spring container using passed configuration class
        // loads beans into container as per configuration class (student1, student2 beans)
    }
}
```

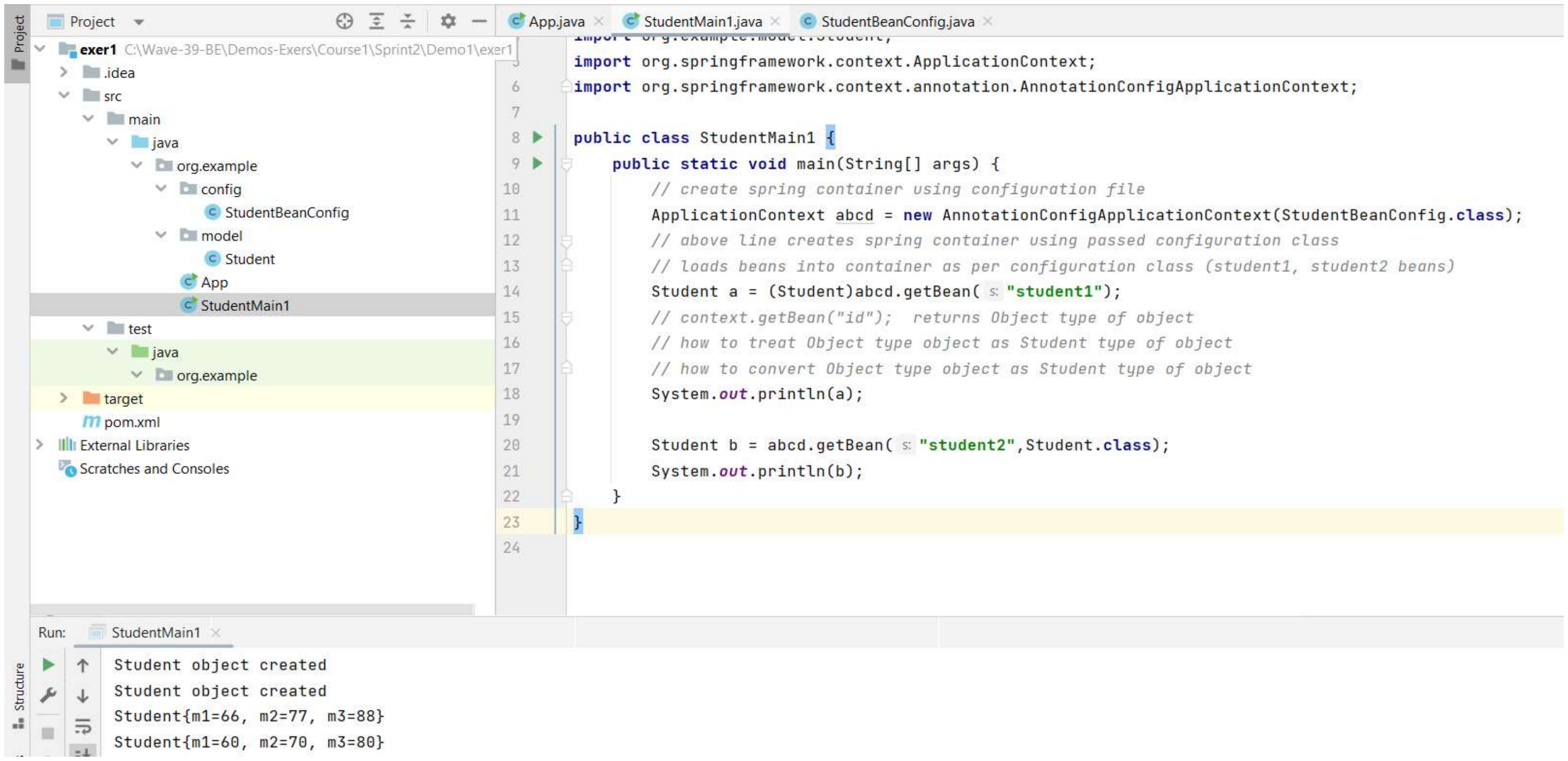
Step 5 by using container ref
get bean and use

parent Object

child Student

how to convert parent type object as child type of object
downcasting





```java
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class StudentMain1 {
    public static void main(String[] args) {
        // create spring container using configuration file
        ApplicationContext abcd = new AnnotationConfigApplicationContext(StudentBeanConfig.class);
        // above line creates spring container using passed configuration class
        // loads beans into container as per configuration class (student1, student2 beans)
        Student a = (Student)abcd.getBean( s: "student1");
        // context.getBean("id");  returns Object type of object
        // how to treat Object type object as Student type of object
        // how to convert Object type object as Student type of object
        System.out.println(a);

        Student b = abcd.getBean( s: "student2",Student.class);
        System.out.println(b);
    }
}
```

```
Student object created
Student object created
Student{m1=66, m2=77, m3=88}
Student{m1=60, m2=70, m3=80}
```

Demo2

pojo : MyTime

int hours, minutes, seconds

Step 1	dependency in pom	done
Step 2	define pojo	done
Step 3	define bean configuration	done
Step 4	main() , create container by loading configuration file	done
Step 5	get beans using container ref and use beans	done

```
class Animal {
        public void move() {
        ….
        }
}

class Dog extends Animal {
// move is derived here
        public void bark() {
        ..
        }
}
```

sorting based on hours

```
MyTime implements Comparable {

        public int compareTo(Object object) {  }


        public boolean equals(Object object) {  }

}
```

```
main()
{
        Animal a = new Animal();
A       a.move();   ?           ok
B       a.bark();   ?           error

        Dog d = new Dog();
C       d.move();   ?           ok
D       d.bark();   ?           ok

        Animal b = new Dog();
E       b.move();   ?           ok
F       b.bark();   ?           error
        ((Dog)b).bark();        ok
}
```

```
Animal b;
        creates object of Animal class
b = new Dog();
        associates / binds Dog properties to b
```

here, b wont get any reference to new methods defined in Dog

`b.bark();    error`

NEED TO COMPLETE 4 EXERs TODAY BY EVENING 5

Student	main1	
	create objects in traditional way	
		parameterized constructor
		setters
	main2	
	create spring container and load beans	
	get beans and use	
MyTime	main1	
	create objects in traditional way	
		parameterized constructor
		setters
	main2	
	create spring container and load beans	
	get beans and use	
Product	Practice	
Movie	Challenge	