Assignment No. 7

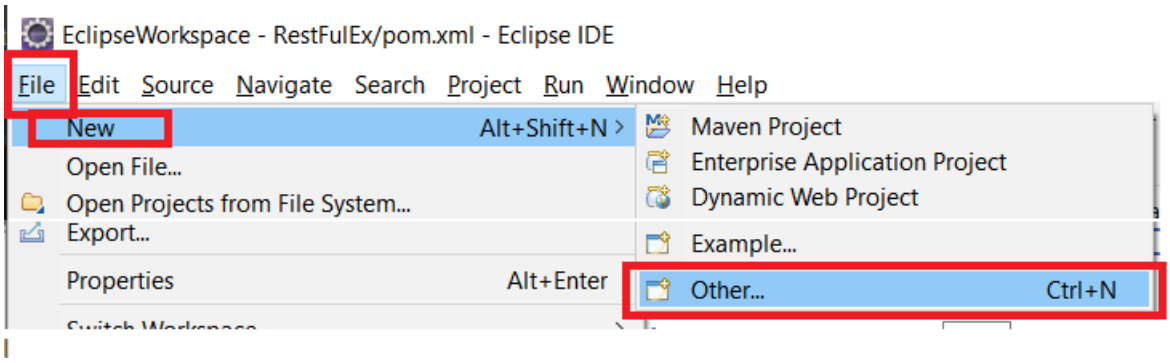
Spring Framework

1. Write a program to print “Hello World” using spring framework.
2. Write a program to demonstrate dependency injection via setter method.
3. Write a program to demonstrate dependency injection via Constructor.

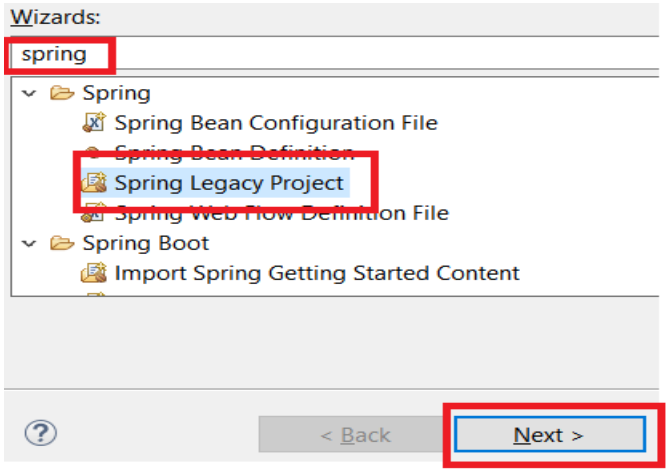
Steps to Create Spring Legacy Project

Step 1 : Creating Spring Legacy Project.

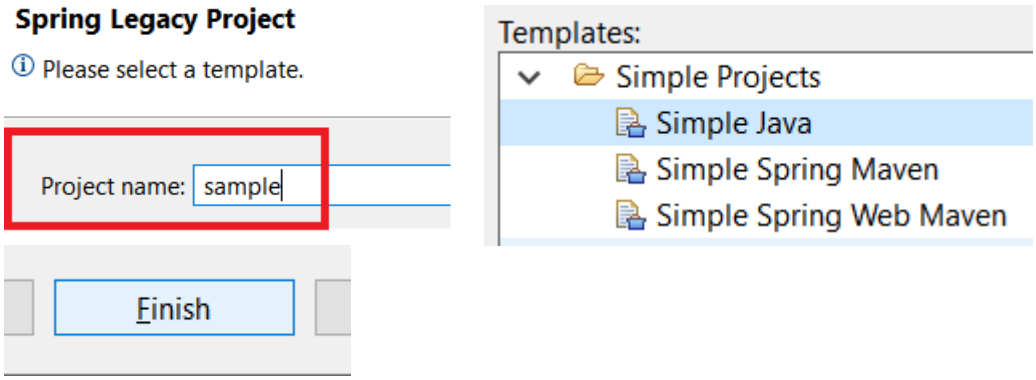
1.1 : Open Eclipse. Go To File > New > Other.



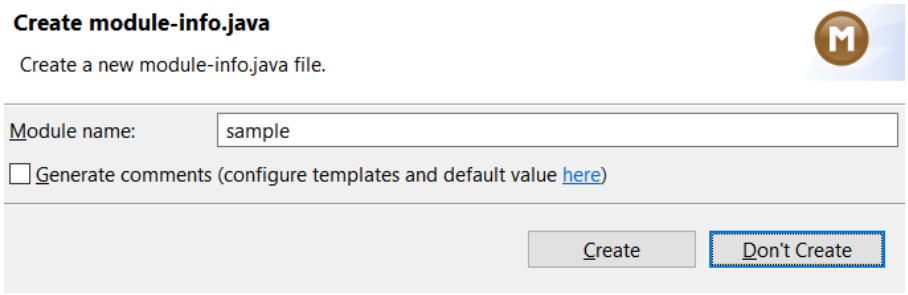
1.2 : Search for ‘spring’ and Select ‘Spring Legacy Project’. Then Click on Next.



1.3 : ChooseProject Name of your wish, below there select **Simple Java** & simply Finish.

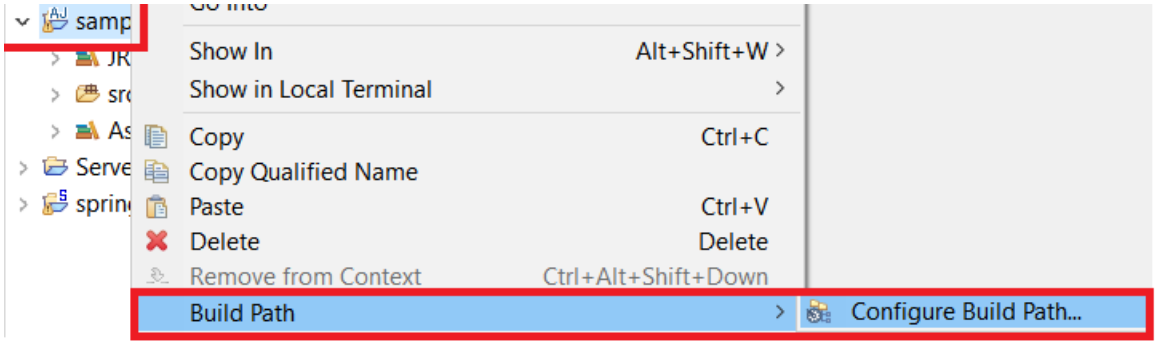


1.4 : If asked for Creating module-info.java file, click on **Don't Create**.

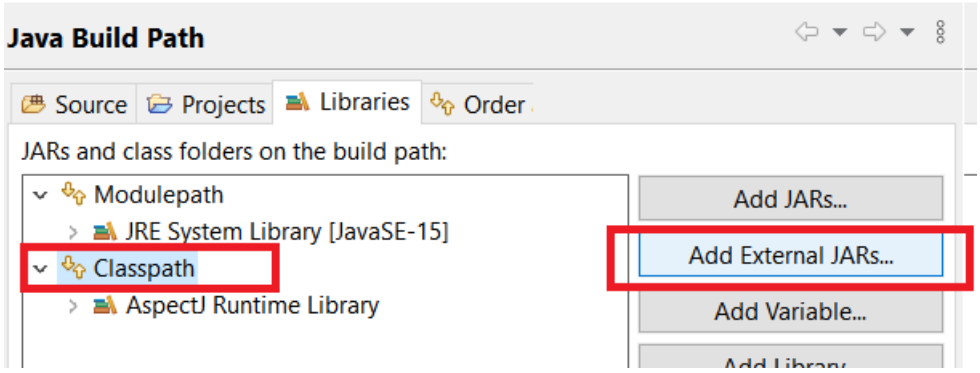


Step 2 : Adding the Spring Libraries.

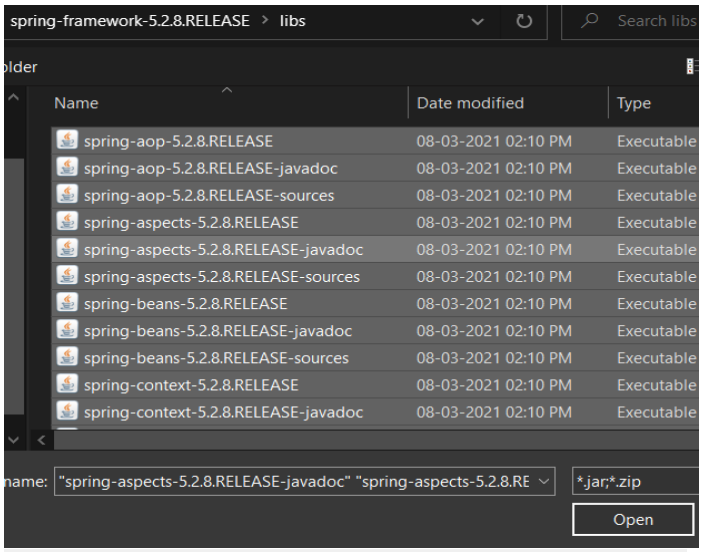
2.1 : Right click on your Newly created Spring Legacy project, Choose Build Path > Configure Build Path.

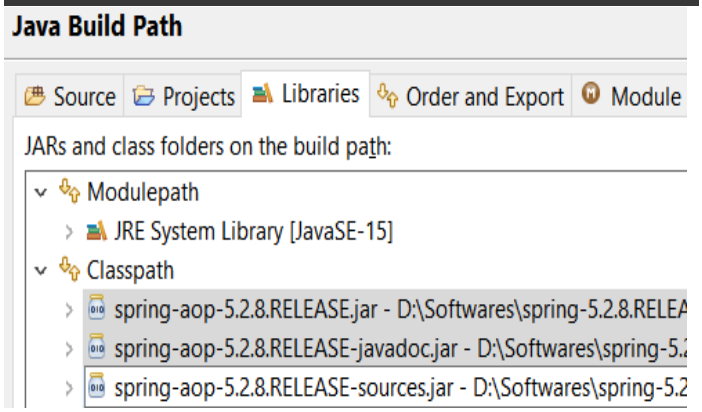


2.2 On Java Build Path wizard, Choose **Classpath** and then select **Add External JARs.**

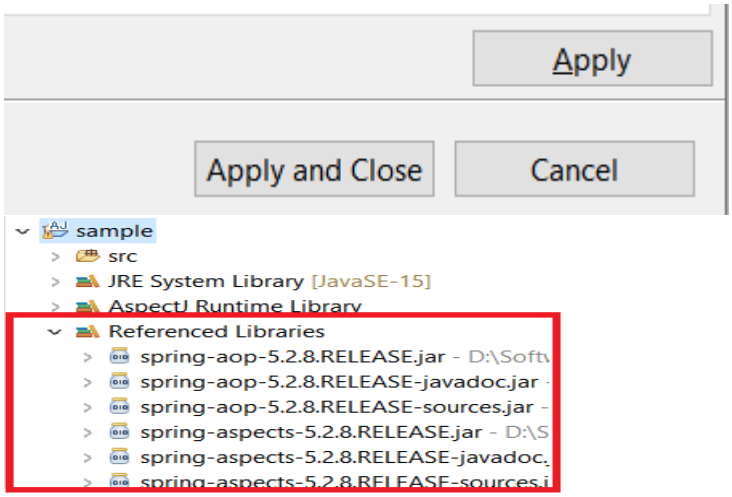


2.3 : Choose all the Spring Libraries you've downloaded, and click on OPEN. This will add all libraries to Classpath.





2.4 Finally click on Apply & Close, now you are ready to work with Spring Legacy Project.



Problem Statement 1 : Write a program to print "Hello World" using spring framework.

Solution :

HelloWorld.java

```
package spring1;

publicclass HelloWorld {

	String name;

	public String getName() {
		return name;
	}

	publicvoid setName(String name) {
		this.name = name;
	}

	@Override
	public String toString() {
		return "Hello World, I'm " + name + ".";
	}
}
```

appctx3.xml

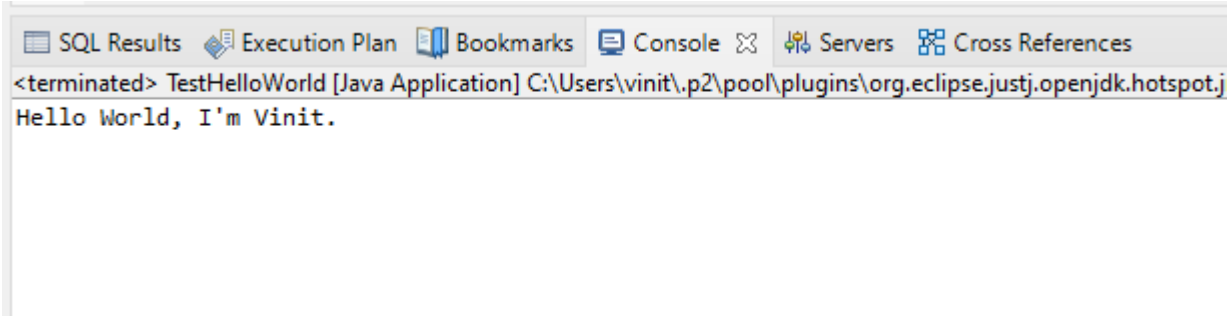
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">

	<bean id="hw" class="spring1.HelloWorld">
		<property name="name"value="Vinit"/>
	</bean>

</beans>
```

TestHelloWorld.java

```
package spring1;

import org.springframework.context.support.ClassPathXmlApplicationContext;

publicclass TestHelloWorld {

	publicstaticvoid main(String[] args) {

		ClassPathXmlApplicationContext app = new
ClassPathXmlApplicationContext("appctx3.xml");
		HelloWorld hw = (HelloWorld) app.getBean("hw");

		System.out.println(hw.toString());

	}
```

}

Output :



Problem Statement 2 : Write a program to demonstrate dependency injection via setter method.

Solution:

Account.java

```
package spring1;

publicclass Account {

	int id;
	String name;
	int balance;

	public Account(int id, String name, int balance) {
		super();
		this.id = id;
		this.name = name;
		this.balance = balance;
	}
	publicint getId() {
		return id;
	}
	publicvoid setId(int id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	publicvoid setName(String name) {
		this.name = name;
	}
	publicint getBalance() {
		return balance;
	}
	publicvoid setBalance(int balance) {
		this.balance = balance;
```

```
	}
	@Override
	public String toString() {
		return "Account [id=" + id + ", name=" + name + ", balance=" + balance + "]";
	}

}
```

appctx2.xml

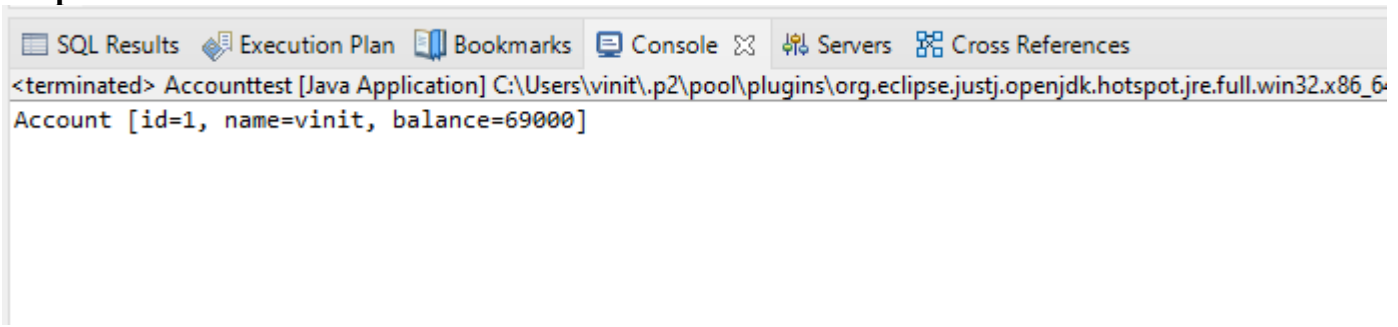
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="Account" class="spring1.Account">

	<constructor-arg name="id" value="1"></constructor-arg>
	<constructor-arg name="name" value="vinit"></constructor-arg>
	<constructor-arg name="balance" value="69000"></constructor-arg>

</bean>

</beans>
```

AccountTest.java

```
package spring1;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Accounttest {

	public static void main(String[] args) {

		ApplicationContext con = new
ClassPathXmlApplicationContext("appctx2.xml");
		Account acc = (Account) con.getBean("Account");
```

```
                System.out.println(acc.toString());

        }
}
```

Output :



Problem Statement 3 : Write a program to demonstrate dependency injection via Constructor.

Solution:

Singer.java

```
package spring1;

publicclass Singer {
        String name;
        int age;
        public String getName() {
                return name;
        }
        publicvoid setName(String name) {
                this.name = name;
        }
        publicint getAge() {
                return age;
        }
        publicvoid setAge(int age) {
                this.age = age;
        }
void displayInfo()
{
        System.out.println("Name:" +name+" Age:" +age);
}
}
```

appctx.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 id="Singer" class="spring1.Singer">
<property name="name" value="vinit"></property>
<property name="age" value="21"></property>
</bean>

</beans>
```

SingerTest.java

```
package spring1;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class SingerTest {

	private static ApplicationContext ctx;
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		ctx=new ClassPathXmlApplicationContext("appctx.xml");
		Singer singer=(Singer)ctx.getBean("Singer");
			singer.displayInfo();



	}

}
```

Output :