00 : 15 : 08
Hours Minutes Seconds

OBJECTIVE QUESTIONS

Intro To Spring

1

Question 1

What is a "Bean" in Spring?

- ○ Component
- ◉ Object
- ○ Class
- ○ Container

Reset Save

00 : 14 : 59
Hours Minutes Seconds

OBJECTIVE QUESTIONS

Spring IoC

1 2 3

Question 1

Consider a CustomerService class in the package com.infosys with appropriate properties and

Which of the following is the CORRECT option to define a bean of CustomerService class in the configuration class?

- ○ @Component
- ◉ @Configuration
- ○ @Bean
- ○ @Service

Reset Save

Spring IoC

1 2 3

Question 2

A Spring standalone application has followed Java configuration to configure its beans is loaded as follows assuming AppConfig.java as the configuration file.

```
ApplicationContext applicationContext = new AnnotationConfigApplicationContex
```

Which of the following statements is/are FALSE with respect to the above-given code

a. Replace AnnotationConfigApplicationContext with ApplicationContext to load the c

b. For Java configuration to work correctly, replace AnnotationConfigApplicationCont AnnotationConfigApplicationContext(AppConfig.java)

- [ ] Only a
- [x] Only b
- [ ] Both a and b
- [ ] Neither a nor b

Reset Save

Spring IoC

1 2 3

Question 3

Identify which among the following statements is/are TRUE with respect to S

a.The Spring IoC Container creates the Java objects that needs to be injecte
b.The Spring IoC Container takes only the metadata file as input to generate
c.The Spring IoC Container takes only the POJO classes as input to generate
d.The Spring IoC Container takes both the metadata file and POJO classes a

- (●) Only a) and d)
- ( ) Only a) and b)
- ( ) Only a)
- ( ) Only d)

Reset Save

Dependency Injection

1 2

}

The main method of the application includes the following statements:

```
AbstractApplicationContext context = new AnnotationConfigApplicationCont
//Line2
bookService.setBookName("Java 9");
System.out.println(bookService.getBookName());
```

Which of the following given option/s can be used in place of Line2 in the

a. BookService bookService = (BookService)context.getBean("bookServic
b. BookService bookService = context.getBean("bookService");
c. BookService bookService = context.getBean("bookService",BookServic

- ○ Only a)
- ○ Only a) and b)
- ◉ Only a) and c)
- ○ Only b)

Reset Save

Dependency Injection

1 2

Question 2

Which of the following is/are the advantages of Dependency Injection(DI)?

a. Loose coupling

b. Re-use of code, ease of testing and maintainable

c. Separation of responsibility by keeping code and configuration separately.

- Only a)
- Only b)
- Only c)
- All the given options

Reset Save

Autowiring

1 2

Question

Analyze the code below. Assume class B and a bean of class B exists in the

```
public class A{
    private B b;
    public A(){ }
    @Autowired
    public void setB(B b){
        this.b=b;
    }
}
```

Which of the following Spring feature is observed in the above given cod

- Spring AOP
- Setter injection
- Constructor Injection
- Spring Auto-scanning

Autowiring

1 2

```
        return addressService;
    }

    @Bean(name = "addr2")
    public AddressService addressService() {
        AddressService addressService = new AddressService();
        addressService.setCity("Mysore");
        return addressService;
    }

    @Bean(value = "customer")
    public CustomerService customerService() {
        return new CustomerService();
    }
}
```

The main method of the application includes the following statements:

```
AbstractApplicationContext context = new AnnotationConfigApplicationCon
CustomerService customer = (CustomerService)context.getBean("custom
System.out.println(customer.getAddressService().getCity());
```

What will be the output, when the above code is executed?

- ○ Bangalore
- ○ Mysore
- ○ BeanInstantiationException: Could not instantiate customer bean
- ◉ Compilation error: Application cannot use both annotation and xml based configura

Bean Scope

1

Questio

John finds the @Scope annotation usage in the code during his code walk
behavior of @Scope. Help him by identifying the RIGHT options from the

a. @Scope can be used as a method-level annotation

b. @Scope can be used as a class level annotation

c. By default, @Scope attribute is set to prototype

d. @Scope has to be used mandatorily while defining bean using @Bean

- ○ Only a)
- ○ Only b)
- ○ Only a) and b)
- ◉ Only d)

Reset Save

00 : 13 : 26
Hours Minutes Seconds

OBJECTIVE QUESTIONS

Autoscan

1 2

Question 1

Which of the below-given options is/are CORRECT with respect to the Auto sc

a.@ComponentScan tag is not required when beans are configured using an

b.@ComponentScan tag is not required when beans are configured using XM

c.@ComponentScan annotation can be used in Java configuration for the aut

- Only b
- Only c
- Only b and c
- All a,b and c

Reset Save

Autoscan

1 2

```
        -----
}
package com.mypack.demo2;
@Component
public class Address{
        -----
}
```

Which of the below given statements is/are VALID in xml configuration to

a) <context:component-scan base-package="com.mypack.demo1" />
b) <context:component-scan base-package="com.mypack.demo2" />
c) <context:component-scan base-package="com" />
d) <context:component-scan base-package="com.mypack" />

- (●) Only c) and d)
- ( ) Only a)
- ( ) Only b)
- ( ) Only c)

Reset Save

Java Based Configuration

1

Consider the below Java configuration file:

```
@Configuration
public class ConfigClass {
  @Bean
  public AddressService addrBean() {
        AddressService addressService = new AddressService("B
        return addressService;
  }
}
```

What is the default bean name of the AddressService class in the above

- ○ AddressService
- ○ addressService
- ◉ addrBean
- ○ addr

Reset Save

OBJECTIVE QUESTIONS

Spring AOP

1 2 3

Below given is the Spring configuration in AppConfig.java :

```
@Configuration
@ComponentScan("com.infy")
//Line2
public class AppConfig {
}
```

The main method of the application includes the following statements:

```
AbstractApplicationContext context = new AnnotationConfigApplicationCont
StudentService stu=(StudentService)context.getBean("studentService");
System.out.println(stu.getName());
```

Identify the CORRECT annotations to be used at Line1 and Line2 of the ab

Before execution of the method

Jack

- (●) @Component at Line1 and @EnableAspectJAutoProxy at Line2
- ( ) @EnableAspectJAutoProxy at Line1 and @Component at Line2
- ( ) @ComponentScan at Line1 and @Component at Line2
- ( ) @Configuration at Line1 and @EnableAspectJAutoProxy at Line2

Reset Save

Spring AOP

1 2 3

```
if (n1==0) {
        System.out.println(" '0' is not valid, c
        n1 = 10;
}
if (n2==0){
        System.out.println(" '0' is not valid,
        n2=10;
}
return joinPoint.proceed(new Object[] { n1,
    }
}
```

Assume the required configuration is already present in SpringConfig

statements:

```
AbstractApplicationContext context = new AnnotationConfigAppl
MyInterface sum = (MyInterface) context.getBean("calculate");
int result = sum.divide(100, 0);
System.out.println("Output is : = " + result);
```

What will be the output, when the above code is executed?

- Output is : = 0
- Runtime exception: Invalid poincut expression
- '0' is not valid, changing number to 10
  Output is : = 10
- Output is : = 10

Spring AOP

2 3

```
}
```

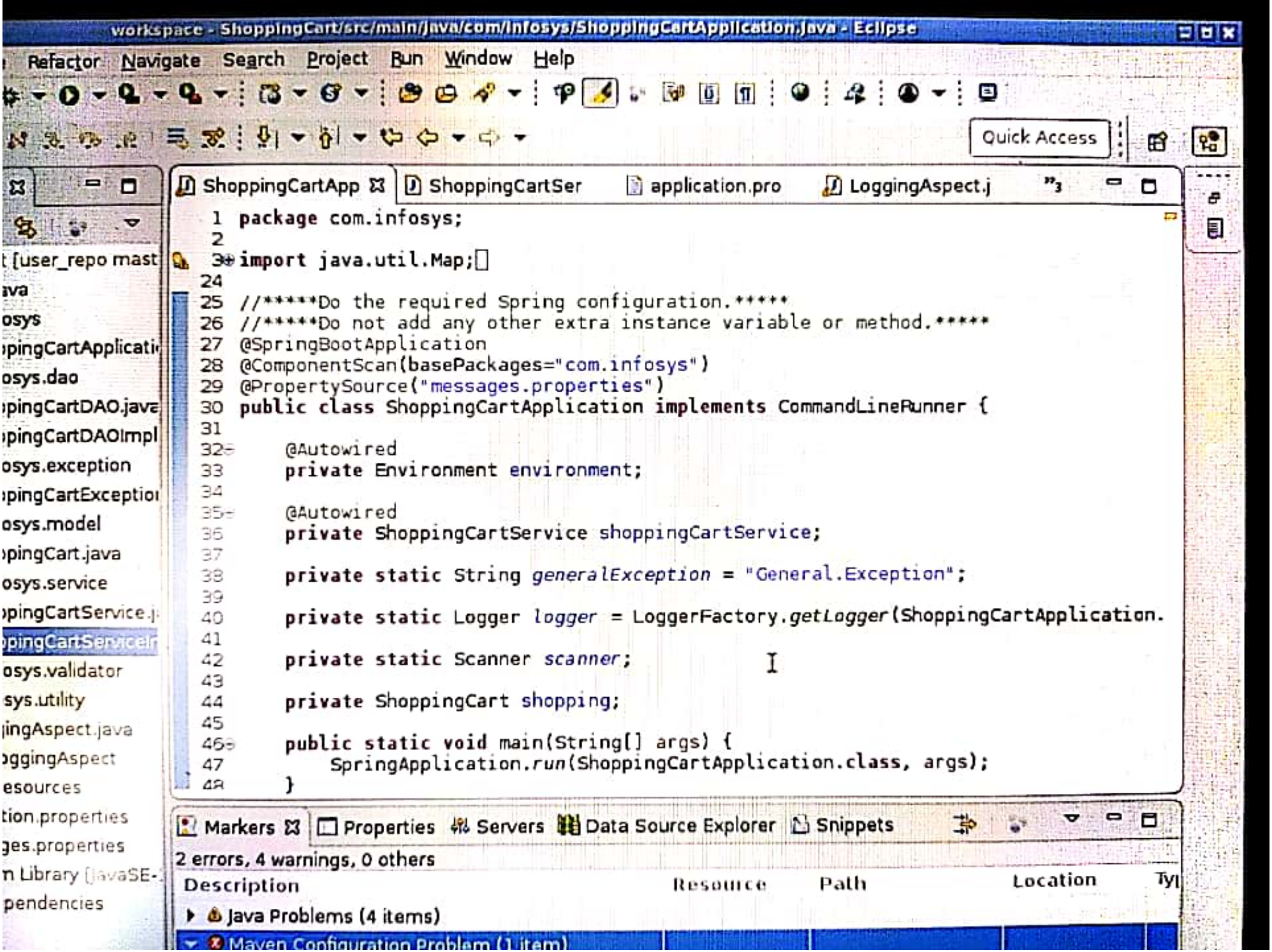
Assume the required configuration is already present in SpringConfiguration.java. The ma
statements:

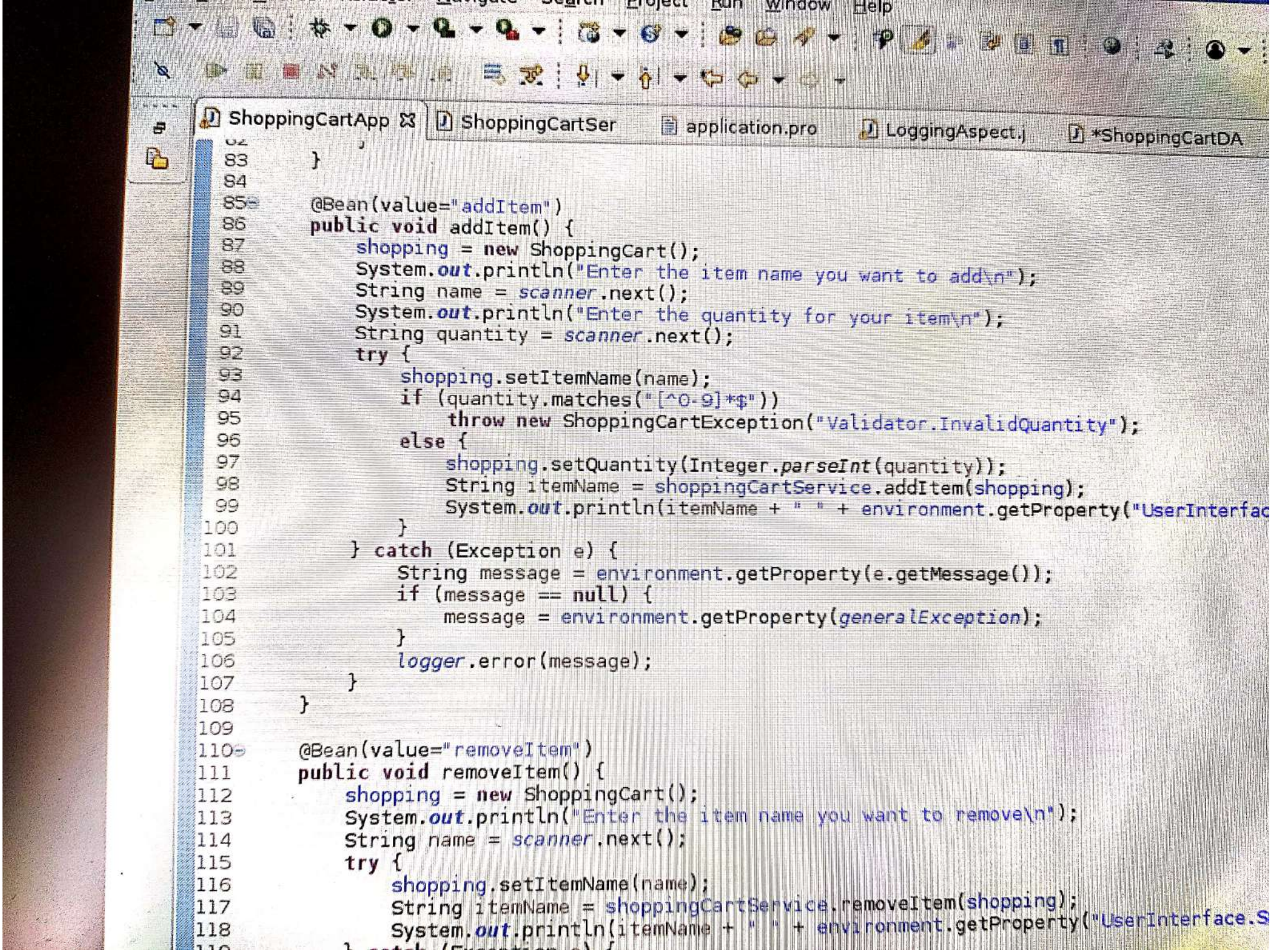
```
AbstractApplicationContext context = new AnnotationConfigApplicationContext(Sprin
Calculate sum = (Calculate) context.getBean("calculate");
System.out.println("Output is : = " + sum.add(100, 10));
System.out.println("Output is : = " + sum.sub(5.5, 2.0));
```

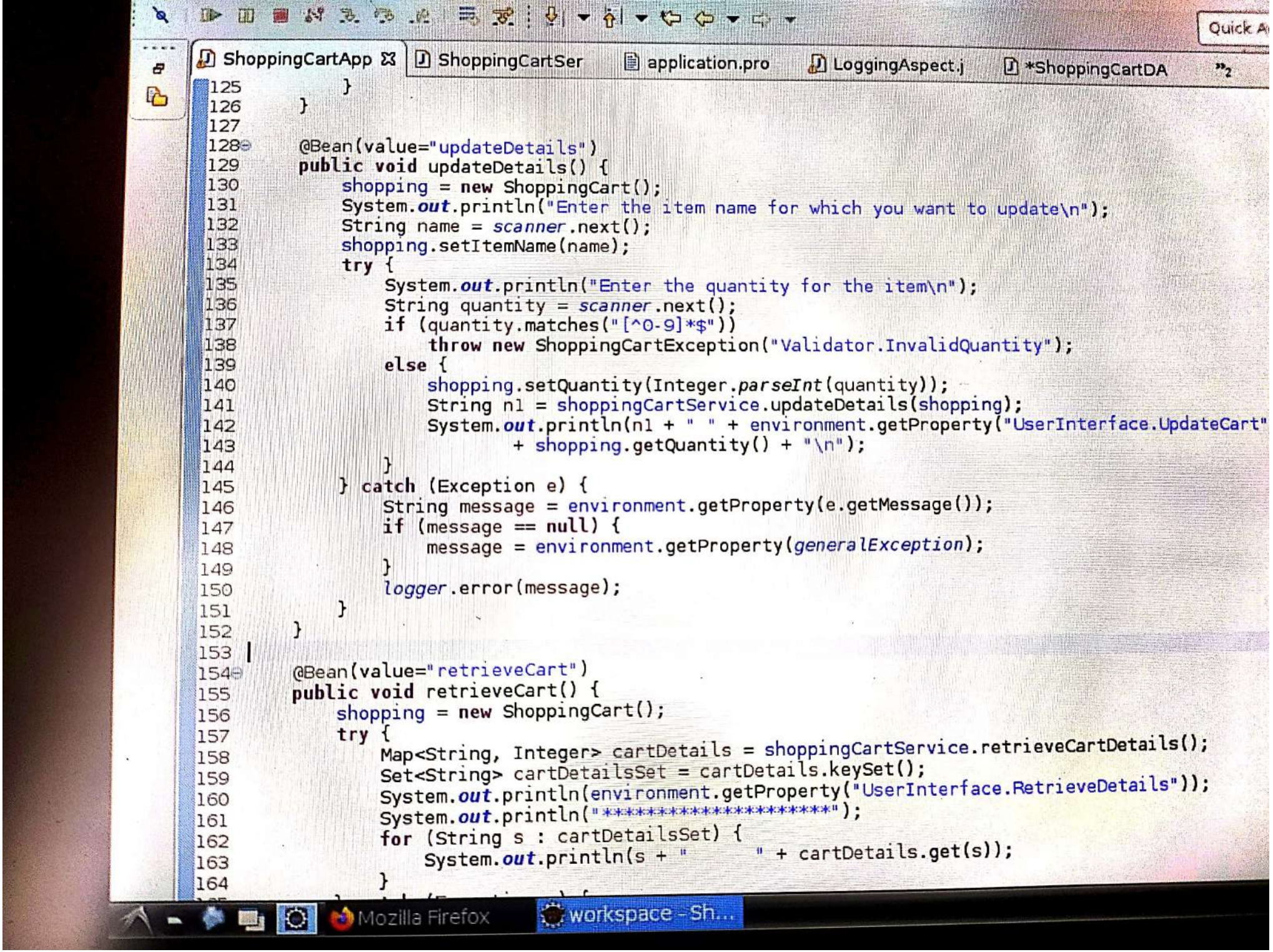
What will be the output, when the above code is executed?

- Output is : = 110
  Output is : = -3.5
- Before Calling the Method.
  Output is : = 110
  Output is : = -3.5
- Before Calling the Method.
  Output is : = 110
  Before Calling the Method.
  Output is : = -3.5
- Output is : = 110
  Before Calling the Method.
  Output is : = -3.5

Reset Save








```
125         }
126     }
127
128     @Bean(value="updateDetails")
129     public void updateDetails() {
130         shopping = new ShoppingCart();
131         System.out.println("Enter the item name for which you want to update\n");
132         String name = scanner.next();
133         shopping.setItemName(name);
134         try {
135             System.out.println("Enter the quantity for the item\n");
136             String quantity = scanner.next();
137             if (quantity.matches("[^0-9]*$"))
138                 throw new ShoppingCartException("Validator.InvalidQuantity");
139             else {
140                 shopping.setQuantity(Integer.parseInt(quantity));
141                 String n1 = shoppingCartService.updateDetails(shopping);
142                 System.out.println(n1 + " " + environment.getProperty("UserInterface.UpdateCart"
143                         + shopping.getQuantity() + "\n");
144             }
145         } catch (Exception e) {
146             String message = environment.getProperty(e.getMessage());
147             if (message == null) {
148                 message = environment.getProperty(generalException);
149             }
150             logger.error(message);
151         }
152     }
153
154     @Bean(value="retrieveCart")
155     public void retrieveCart() {
156         shopping = new ShoppingCart();
157         try {
158             Map<String, Integer> cartDetails = shoppingCartService.retrieveCartDetails();
159             Set<String> cartDetailsSet = cartDetails.keySet();
160             System.out.println(environment.getProperty("UserInterface.RetrieveDetails"));
161             System.out.println("**********************");
162             for (String s : cartDetailsSet) {
163                 System.out.println(s + "      " + cartDetails.get(s));
164             }
```