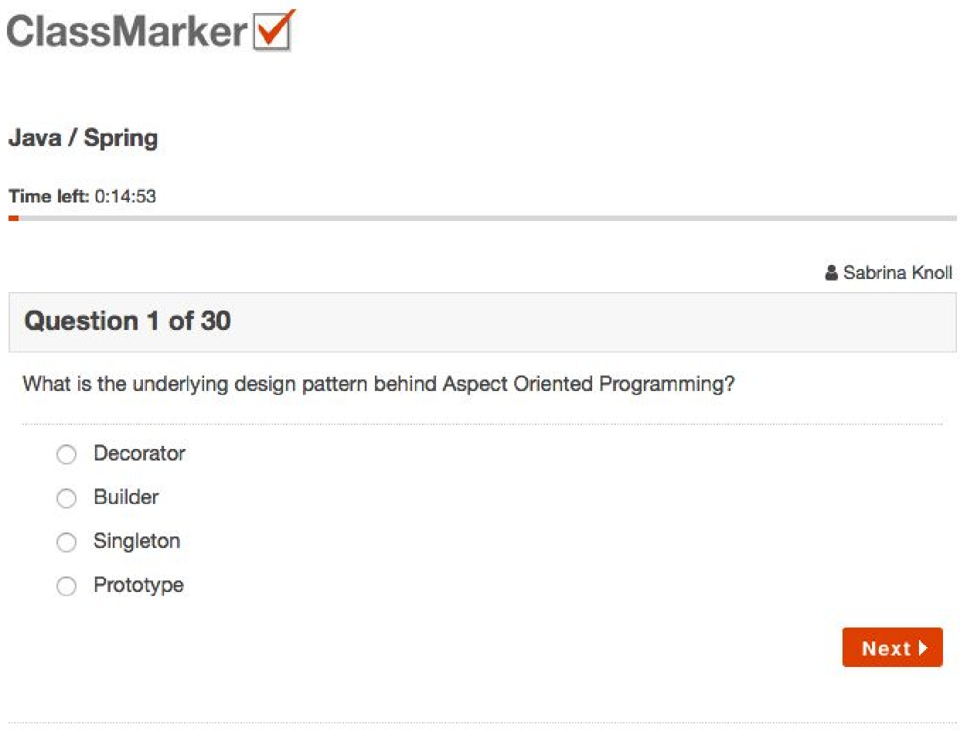
EDGELAB INTERVIEW



Java / Spring

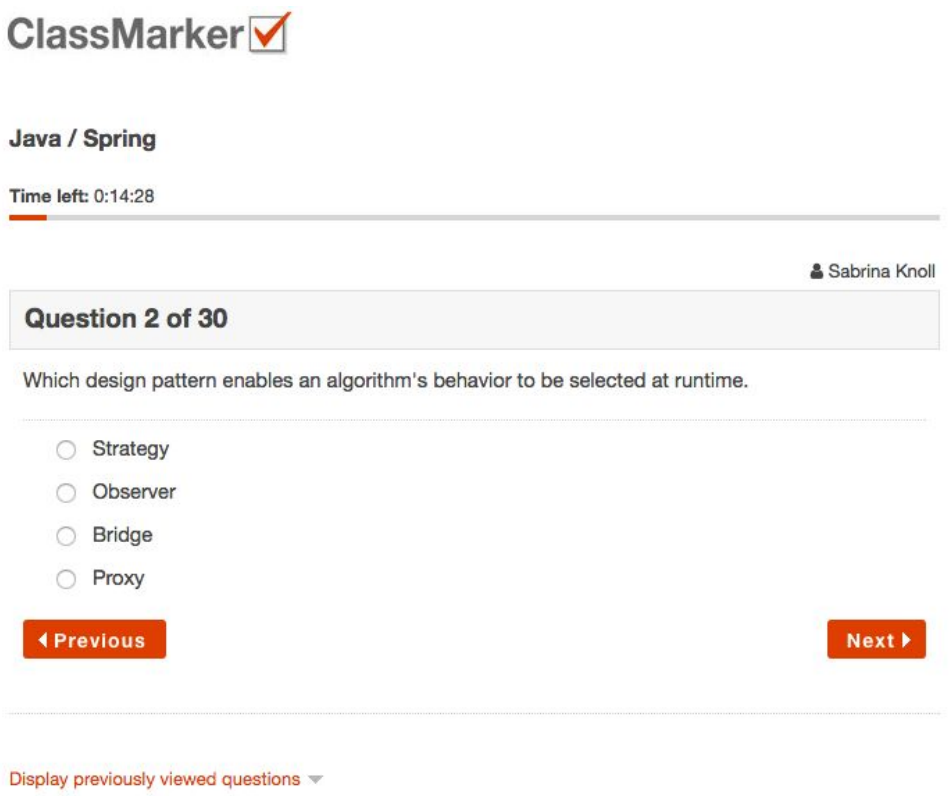
Time left: 0:14:53

Sabrina Knoll

Question 1 of 30

What is the underlying design pattern behind Aspect Oriented Programming?

- Decorator
- Builder
- Singleton
- Prototype



ClassMarker

Java / Spring

Time left: 0:14:28

Sabrina Knoll

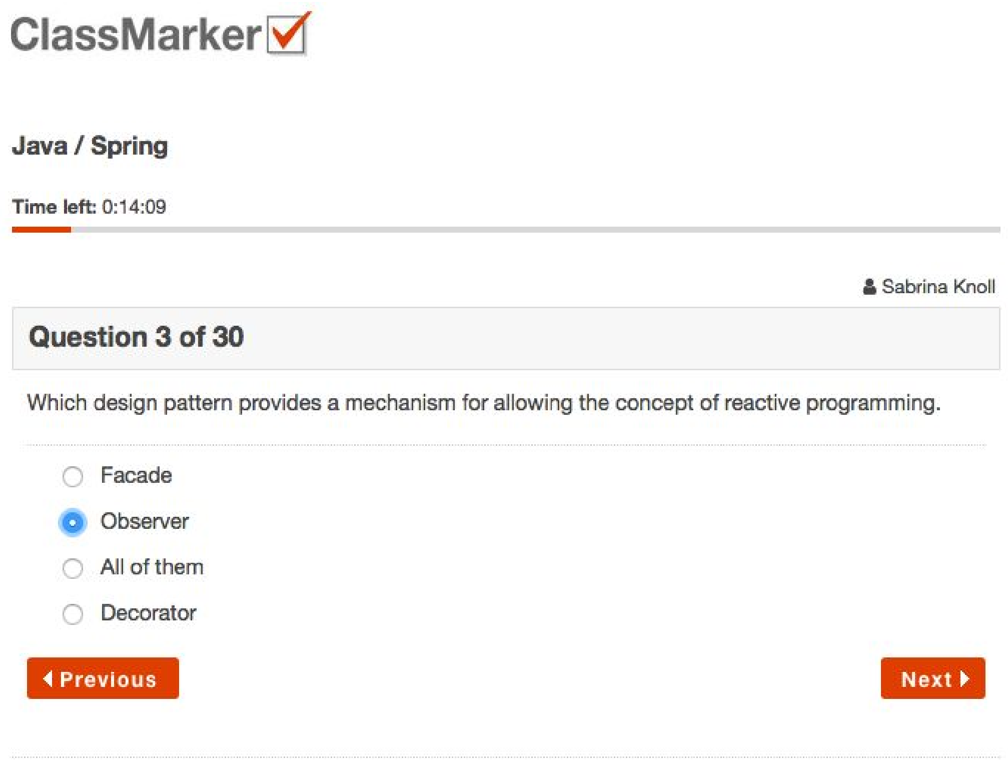
Question 2 of 30

Which design pattern enables an algorithm's behavior to be selected at runtime.

- Strategy
- Observer
- Bridge
- Proxy

Previous | Next

Display previously viewed questions



ClassMarker

Java / Spring

Time left: 0:14:09

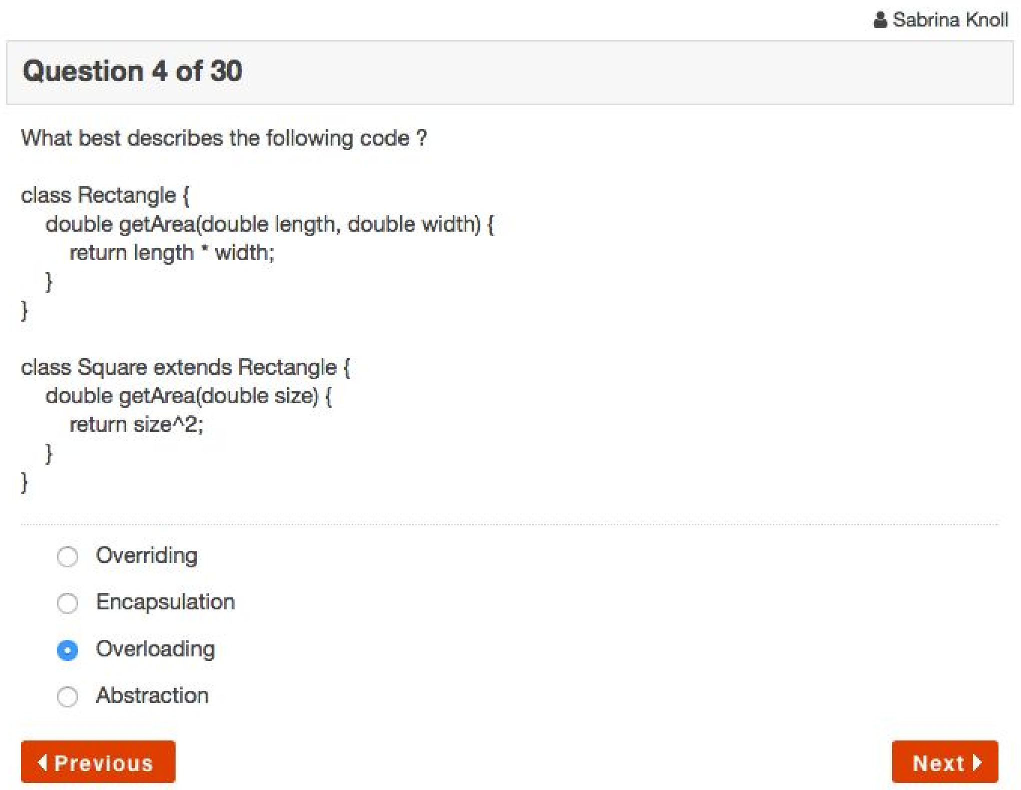
Sabrina Knoll

Question 3 of 30

Which design pattern provides a mechanism for allowing the concept of reactive programming.

- Facade
- [✓ Observer]
- All of them
- Decorator

Previous | Next


Sabrina Knoll

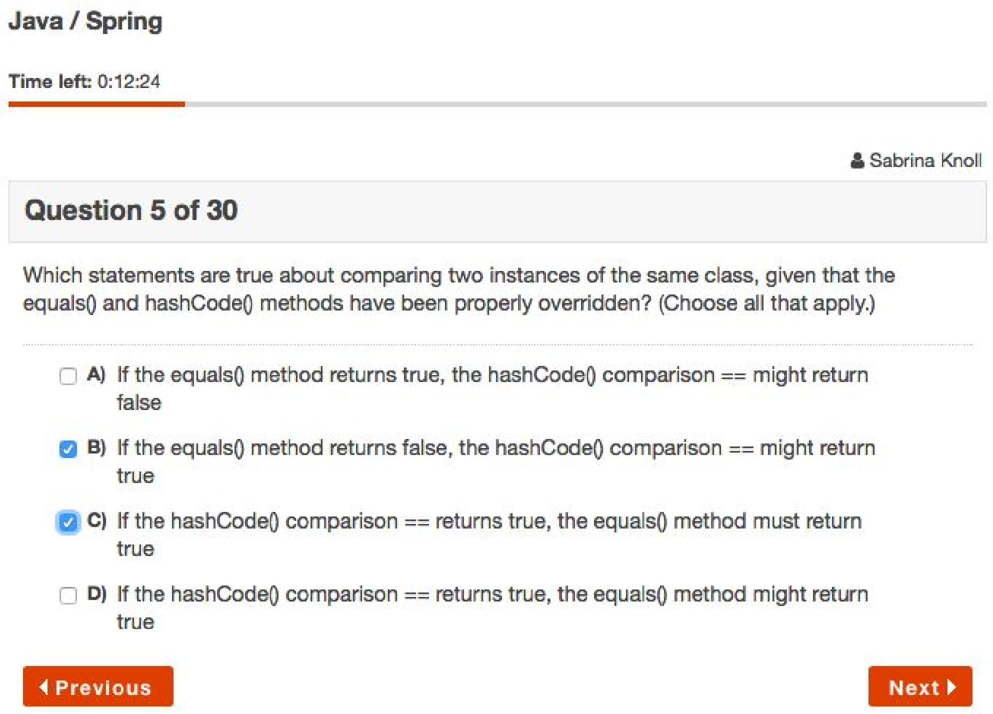
Question 4 of 30

What best describes the following code ?

```
class Rectangle {
   double getArea(double length, double width) {
      return length * width;
   }
}

class Square extends Rectangle {
   double getArea(double size) {
      return size^2;
   }
}
```

- ○ Overriding
- ○ Encapsulation
- [✓ Overloading]
- ○ Abstraction


Java / Spring

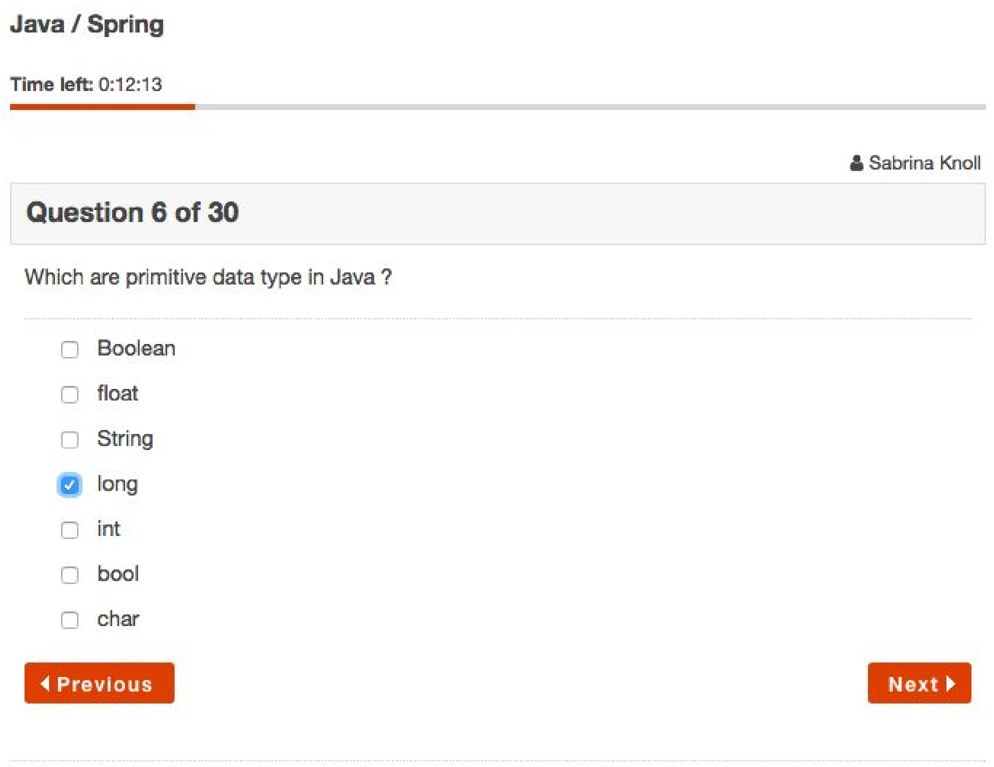
Time left: 0:12:24

Sabrina Knoll

Question 5 of 30

Which statements are true about comparing two instances of the same class, given that the equals() and hashCode() methods have been properly overridden? (Choose all that apply.)

- ☐ **A)** If the equals() method returns true, the hashCode() comparison == might return false
- [✓ **B)** If the equals() method returns false, the hashCode() comparison == might return true]
- [✓ **C)** If the hashCode() comparison == returns true, the equals() method must return true]
- ☐ **D)** If the hashCode() comparison == returns true, the equals() method might return true



Java / Spring

Time left: 0:12:13

Sabrina Knoll

Question 6 of 30

Which are primitive data type in Java ?

- Boolean
- float
- String
- [✓ long]
- int
- bool
- char

Java / Spring

Time left: 0:11:58

Sabrina Knoll

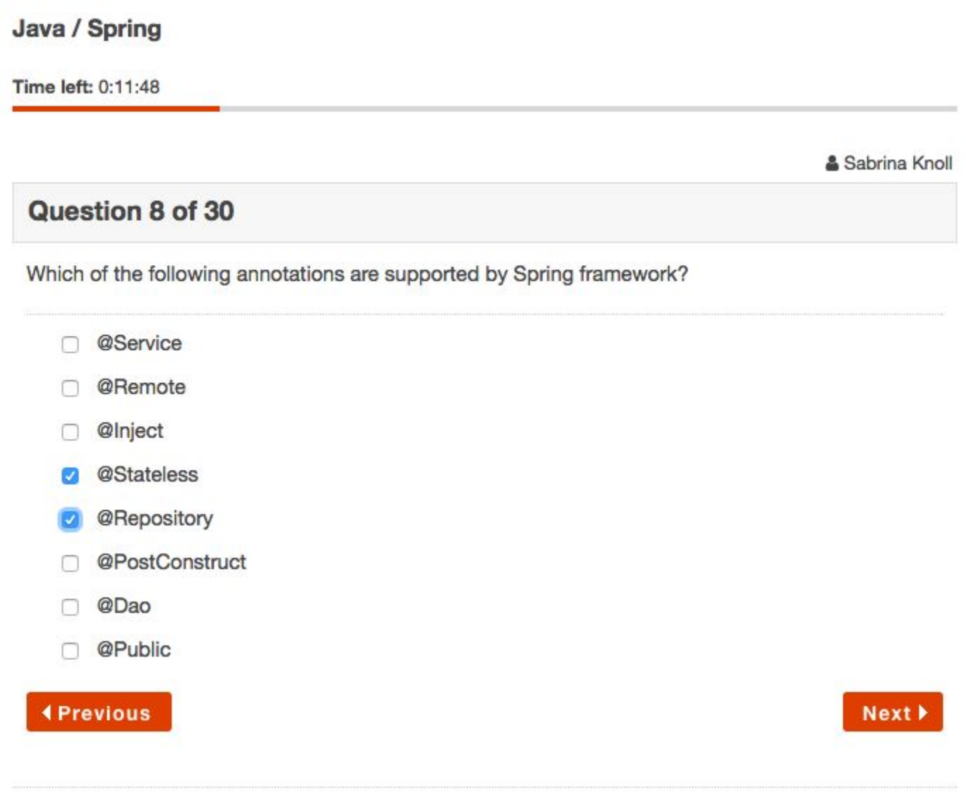
Question 7 of 30

Please fill in the following:

```
public __________ Test {
   private int value1;
   private int value2;
   abstract int add (int v1, int v2);
}
```

- ○ interface
- [✓ abstract interface]
- ○ final class
- ○ abstract class

Previous   Next



Java / Spring

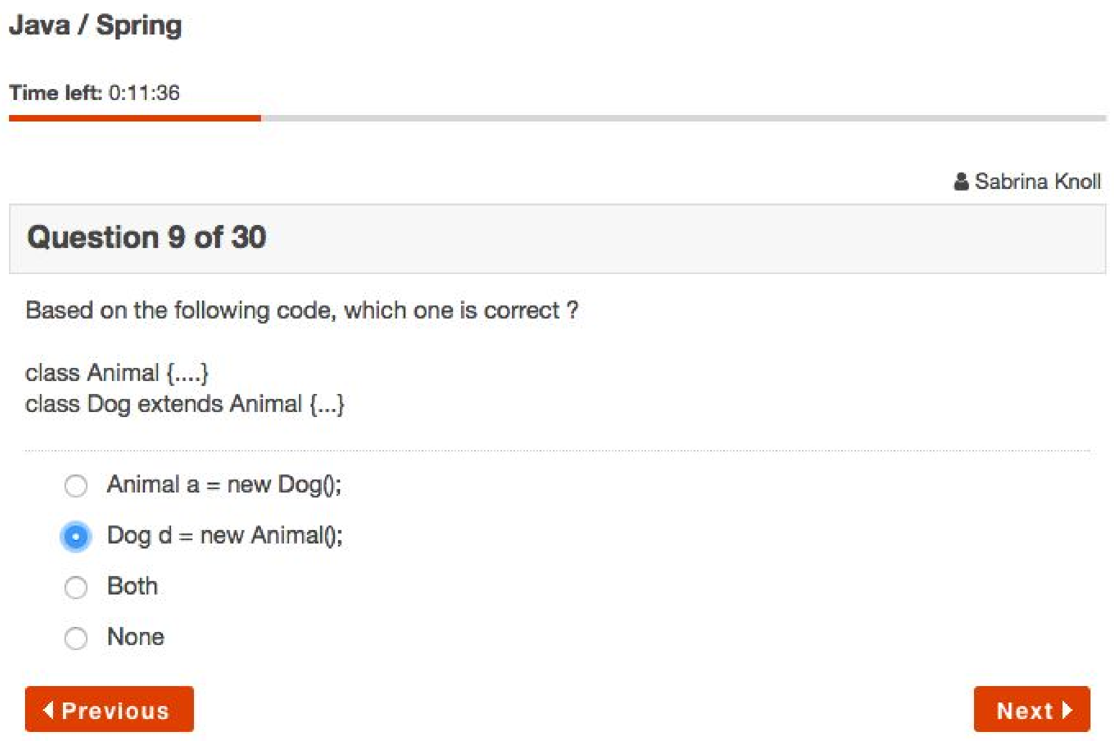
Time left: 0:11:48

Sabrina Knoll

Question 8 of 30

Which of the following annotations are supported by Spring framework?

- @Service
- @Remote
- @Inject
- [✓ @Stateless]
- [✓ @Repository]
- @PostConstruct
- @Dao
- @Public



Java / Spring

Time left: 0:11:36

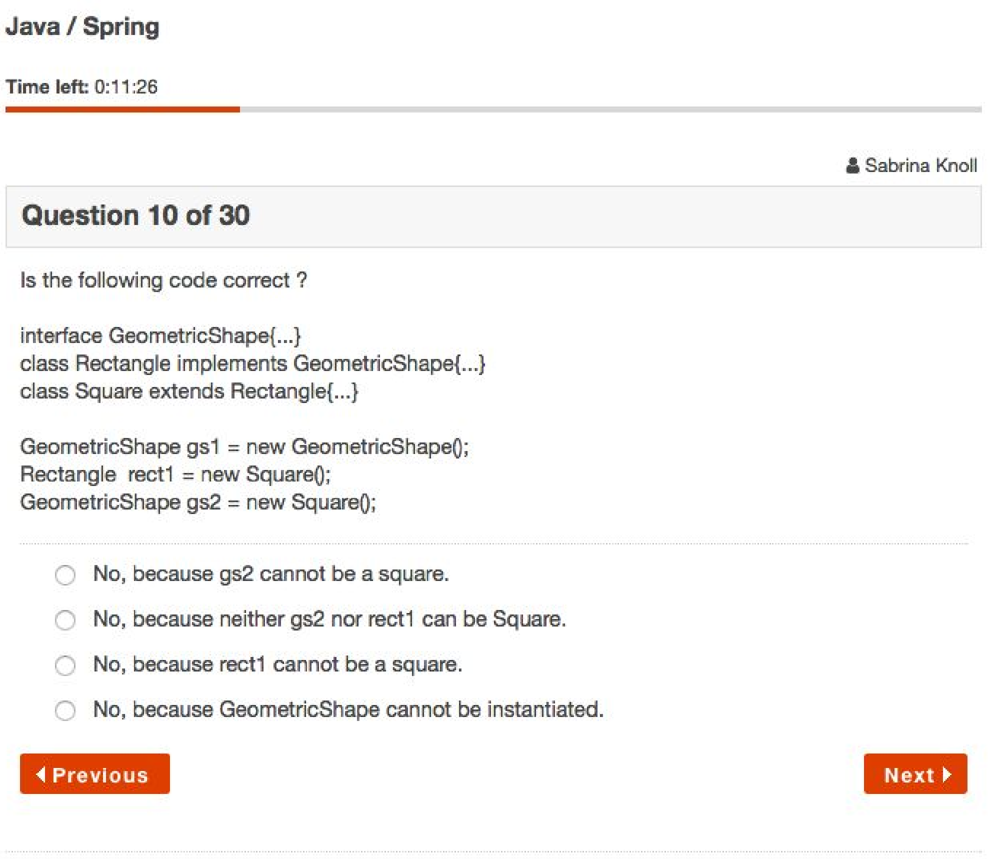
Sabrina Knoll

Question 9 of 30

Based on the following code, which one is correct ?

```
class Animal {....}
class Dog extends Animal {...}
```

- Animal a = new Dog();
- [✓ Dog d = new Animal();]
- Both
- None



Java / Spring

Time left: 0:11:26

Sabrina Knoll

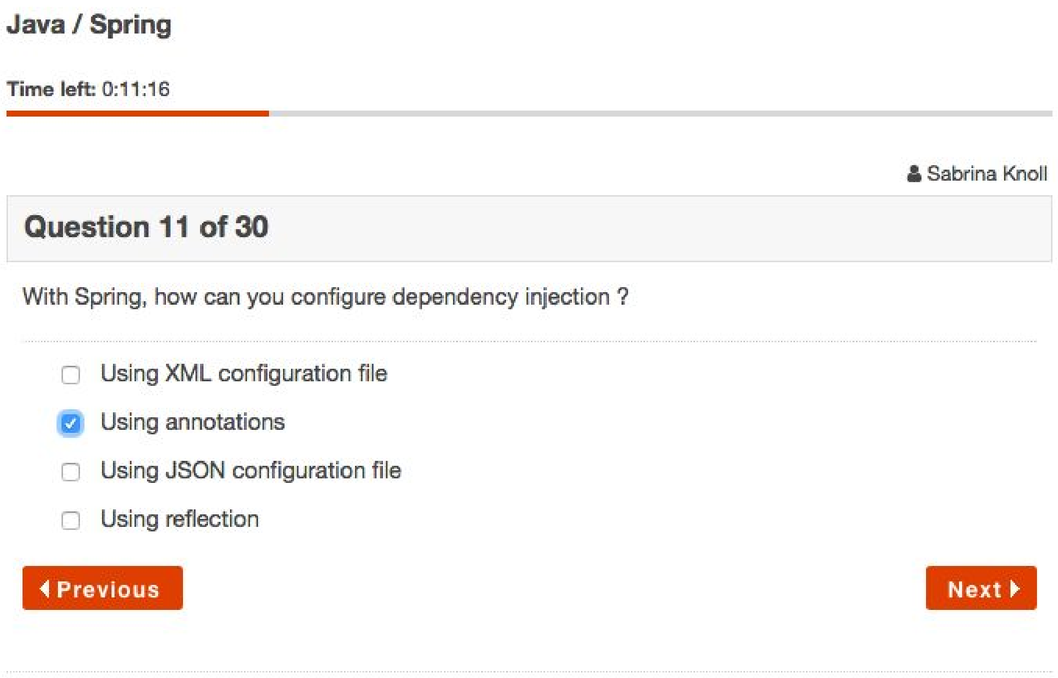
Question 10 of 30

Is the following code correct ?

```
interface GeometricShape{...}
class Rectangle implements GeometricShape{...}
class Square extends Rectangle{...}

GeometricShape gs1 = new GeometricShape();
Rectangle  rect1 = new Square();
GeometricShape gs2 = new Square();
```

- ○ No, because gs2 cannot be a square.
- ○ No, because neither gs2 nor rect1 can be Square.
- ○ No, because rect1 cannot be a square.
- ○ No, because GeometricShape cannot be instantiated.



Java / Spring

Time left: 0:11:16

Sabrina Knoll

Question 11 of 30

With Spring, how can you configure dependency injection ?

- ☐ Using XML configuration file
- [✓ Using annotations]
- ☐ Using JSON configuration file
- ☐ Using reflection

Time left: 0:11:09

Sabrina Knoll

Question 12 of 30

Is this object instantiated when constructor is called ?

```
class Test {
   String value;

   Test() {
      value = null;
   }

   String getValue(){
      return value;
   }
}
```

- No, because value is not initialized when declared
- No, because getValue will return null
- No, because value set to NULL
- Yes

Previous　　Next

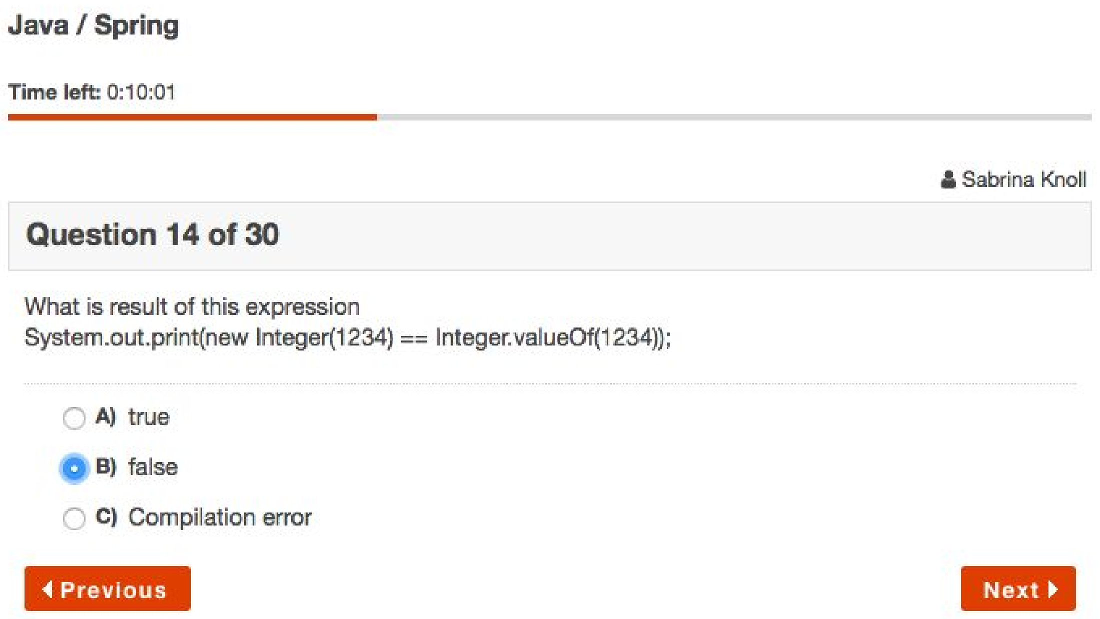
Time left: 0:10:56

Sabrina Knoll

Question 13 of 30

Which method reference could simplify this lambda:
(Foo foo) -> foo.getName()

Previous　　Next



Java / Spring

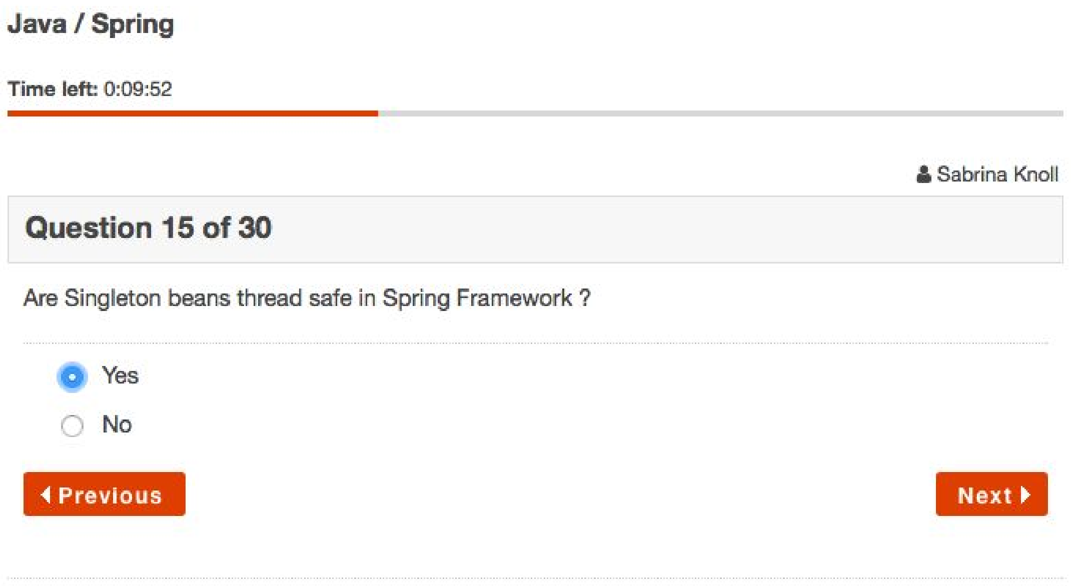
Time left: 0:10:01

Sabrina Knoll

Question 14 of 30

What is result of this expression
System.out.print(new Integer(1234) == Integer.valueOf(1234));

A) true
[✓ B) false]
C) Compilation error



Java / Spring

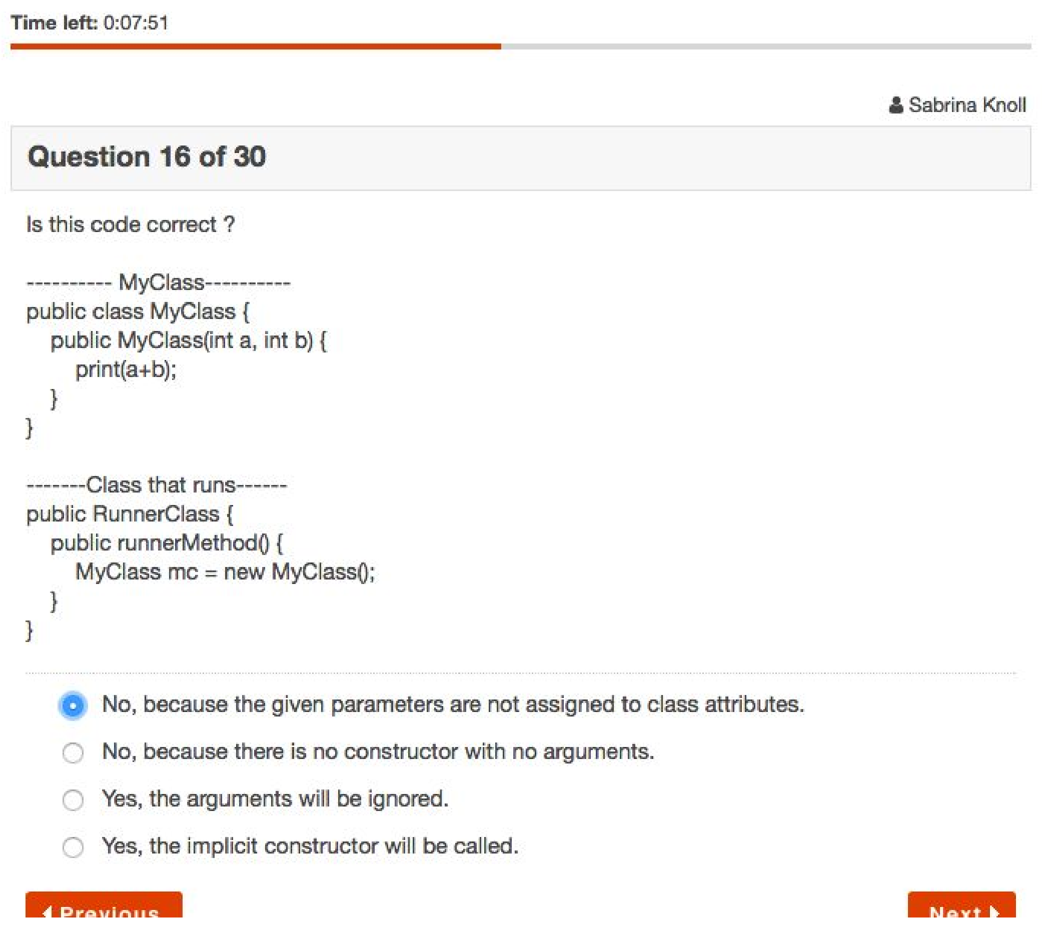
Time left: 0:09:52

Sabrina Knoll

Question 15 of 30

Are Singleton beans thread safe in Spring Framework ?

[✓ Yes]
No



Time left: 0:07:51

Sabrina Knoll

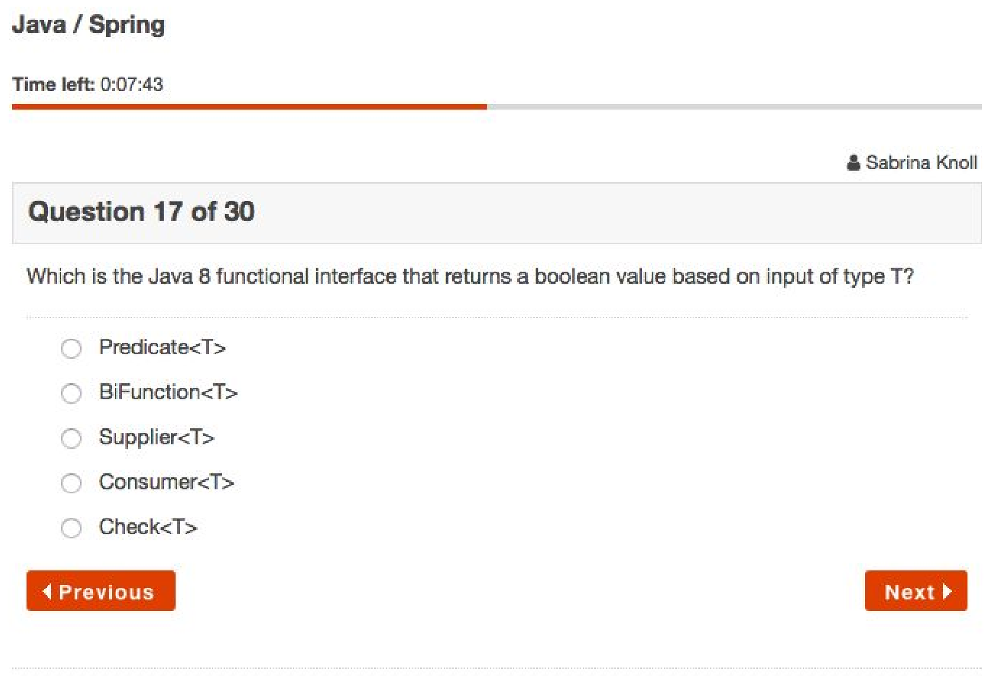
Question 16 of 30

Is this code correct ?

```
---------- MyClass----------
public class MyClass {
   public MyClass(int a, int b) {
      print(a+b);
   }
}

-------Class that runs------
public RunnerClass {
   public runnerMethod() {
      MyClass mc = new MyClass();
   }
}
```

- [✓ No, because the given parameters are not assigned to class attributes.]
- No, because there is no constructor with no arguments.
- Yes, the arguments will be ignored.
- Yes, the implicit constructor will be called.



Java / Spring

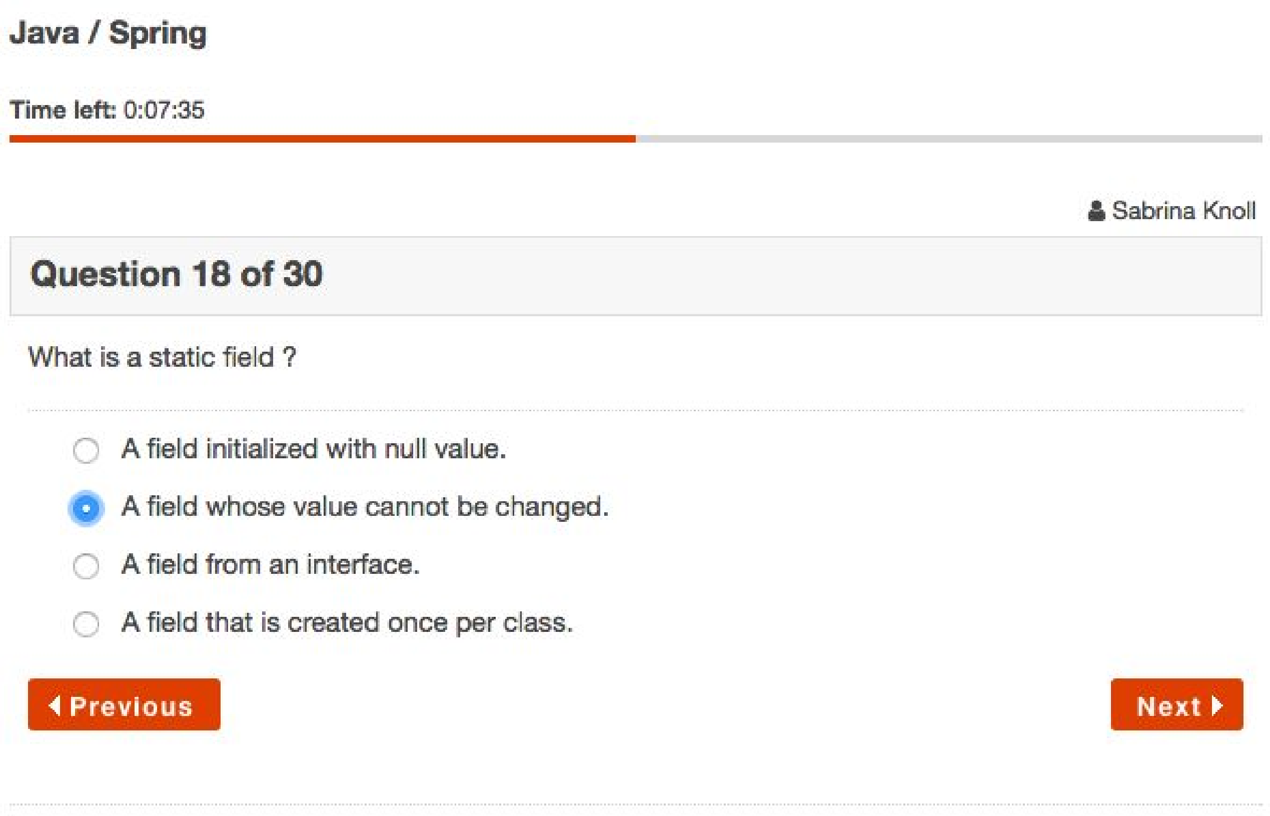
Time left: 0:07:43

Sabrina Knoll

Question 17 of 30

Which is the Java 8 functional interface that returns a boolean value based on input of type T?

- Predicate<T>
- BiFunction<T>
- Supplier<T>
- Consumer<T>
- Check<T>



Java / Spring

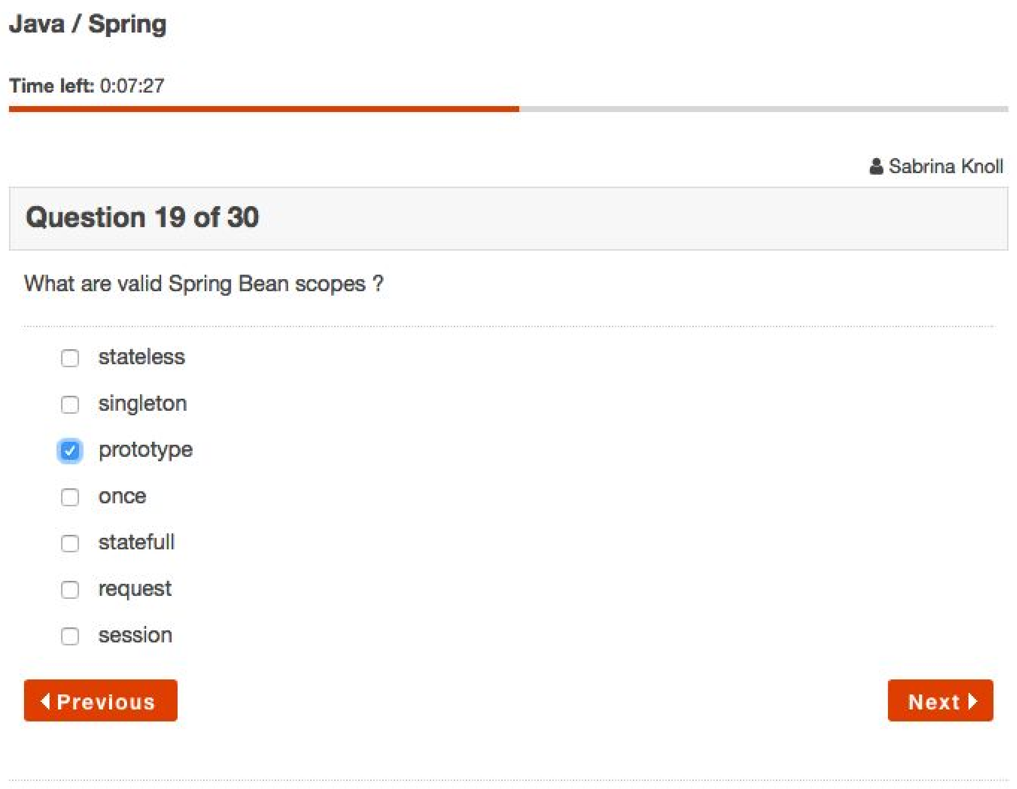
Time left: 0:07:35

Sabrina Knoll

Question 18 of 30

What is a static field ?

- ○ A field initialized with null value.
- [✓ A field whose value cannot be changed.]
- ○ A field from an interface.
- ○ A field that is created once per class.



Java / Spring

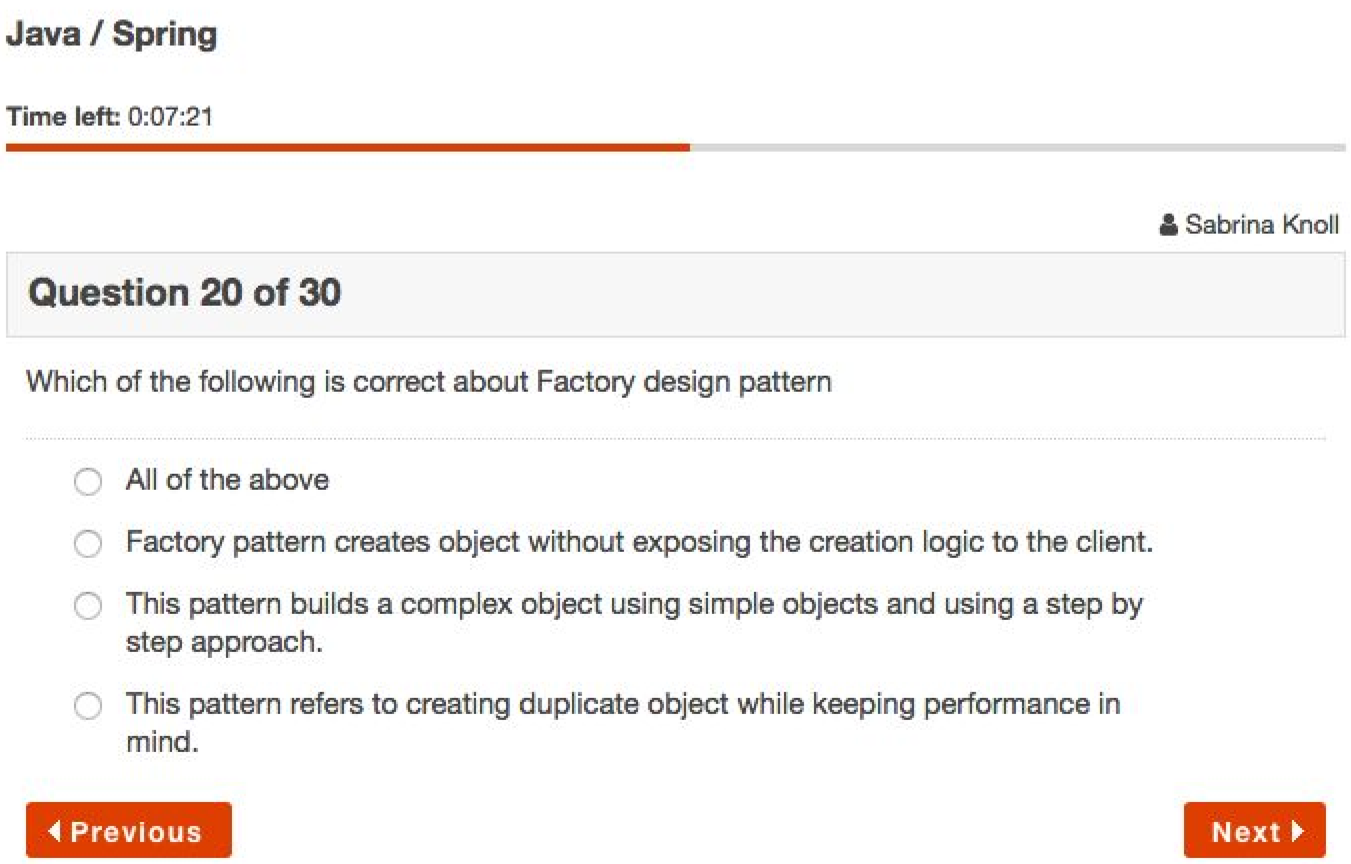
Time left: 0:07:27

Sabrina Knoll

Question 19 of 30

What are valid Spring Bean scopes ?

- ☐ stateless
- ☐ singleton
- ☑ prototype
- ☐ once
- ☐ statefull
- ☐ request
- ☐ session



Java / Spring

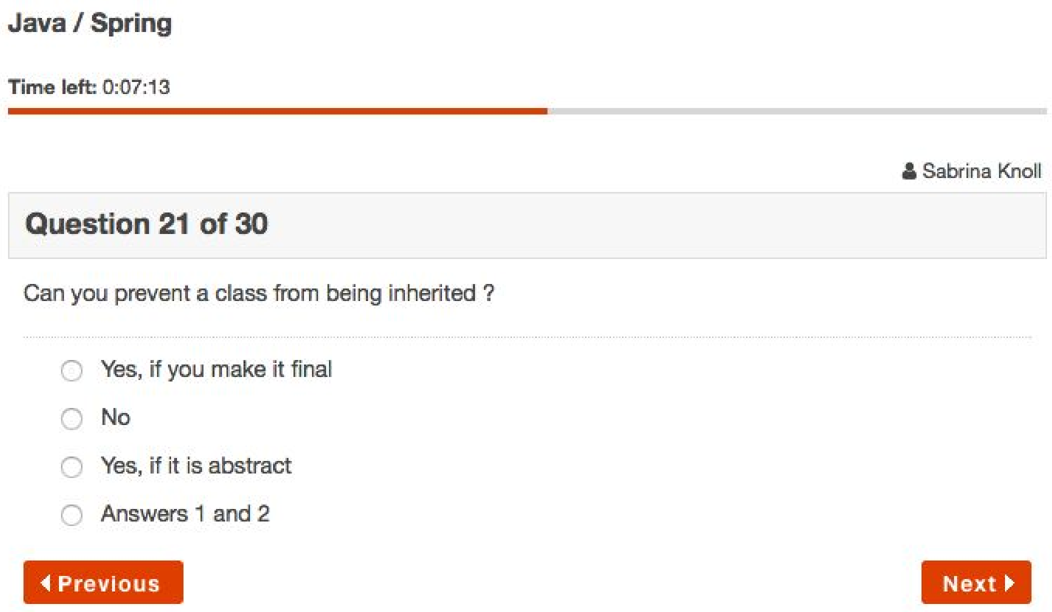
Time left: 0:07:21

Sabrina Knoll

Question 20 of 30

Which of the following is correct about Factory design pattern

- All of the above
- Factory pattern creates object without exposing the creation logic to the client.
- This pattern builds a complex object using simple objects and using a step by step approach.
- This pattern refers to creating duplicate object while keeping performance in mind.



Java / Spring

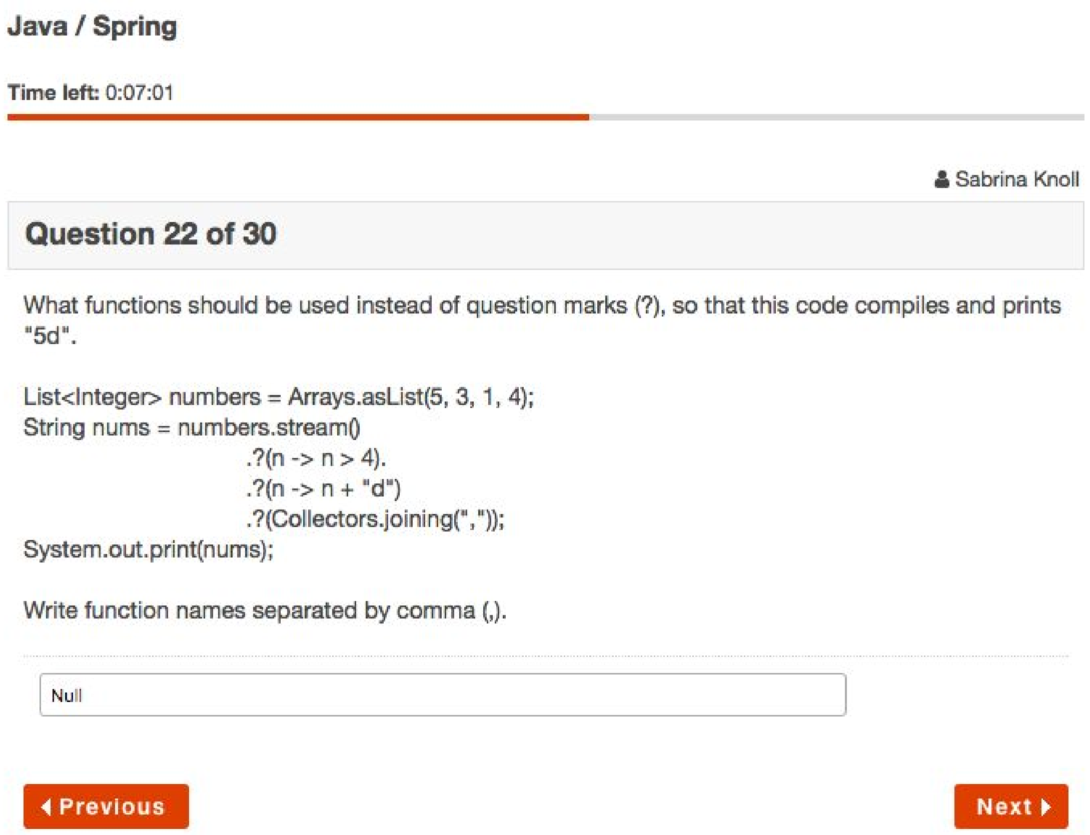
Time left: 0:07:13

Sabrina Knoll

Question 21 of 30

Can you prevent a class from being inherited ?

- Yes, if you make it final
- No
- Yes, if it is abstract
- Answers 1 and 2



Java / Spring

Time left: 0:07:01

Sabrina Knoll

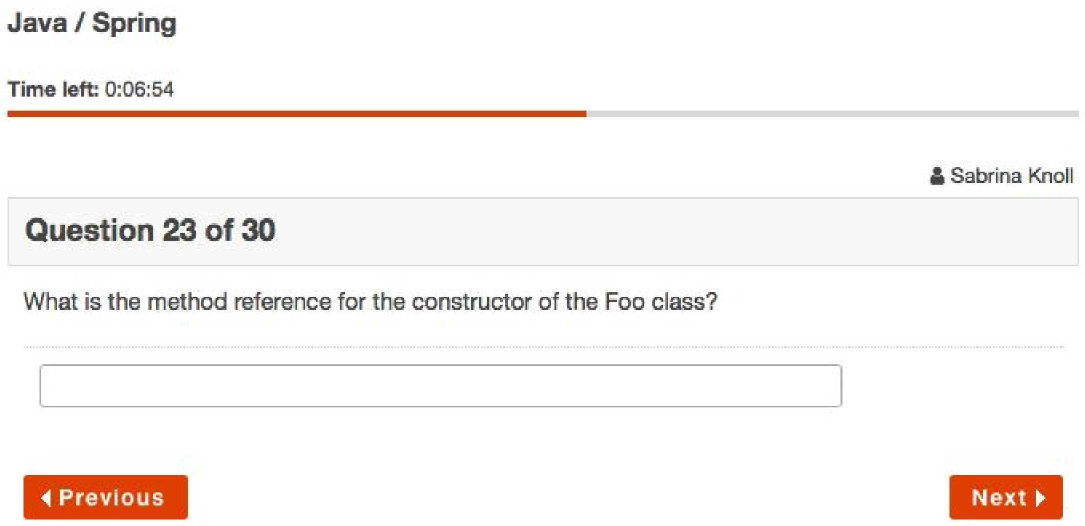
Question 22 of 30

What functions should be used instead of question marks (?), so that this code compiles and prints "5d".

```
List<Integer> numbers = Arrays.asList(5, 3, 1, 4);
String nums = numbers.stream()
                        .?(n -> n > 4).
                        .?(n -> n + "d")
                        .?(Collectors.joining(","));
System.out.print(nums);
```

Write function names separated by comma (,).

Null



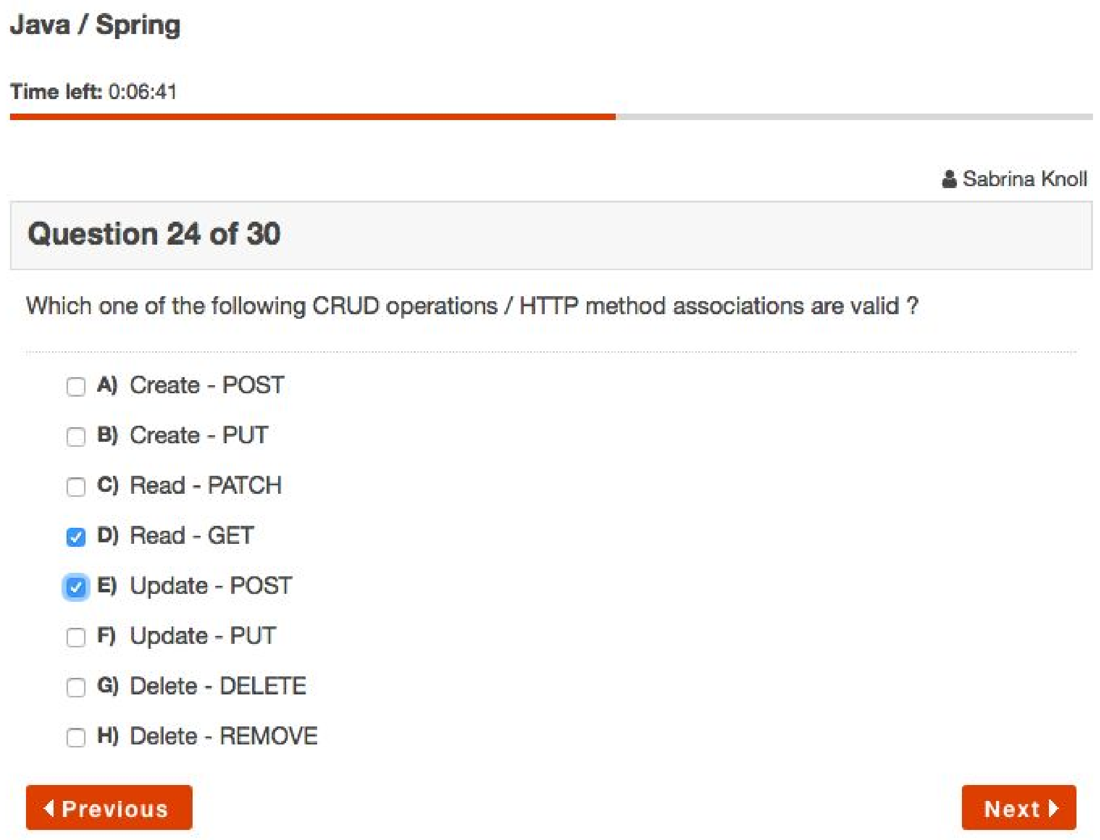
Java / Spring

Time left: 0:06:54

Sabrina Knoll

Question 23 of 30

What is the method reference for the constructor of the Foo class?



Java / Spring

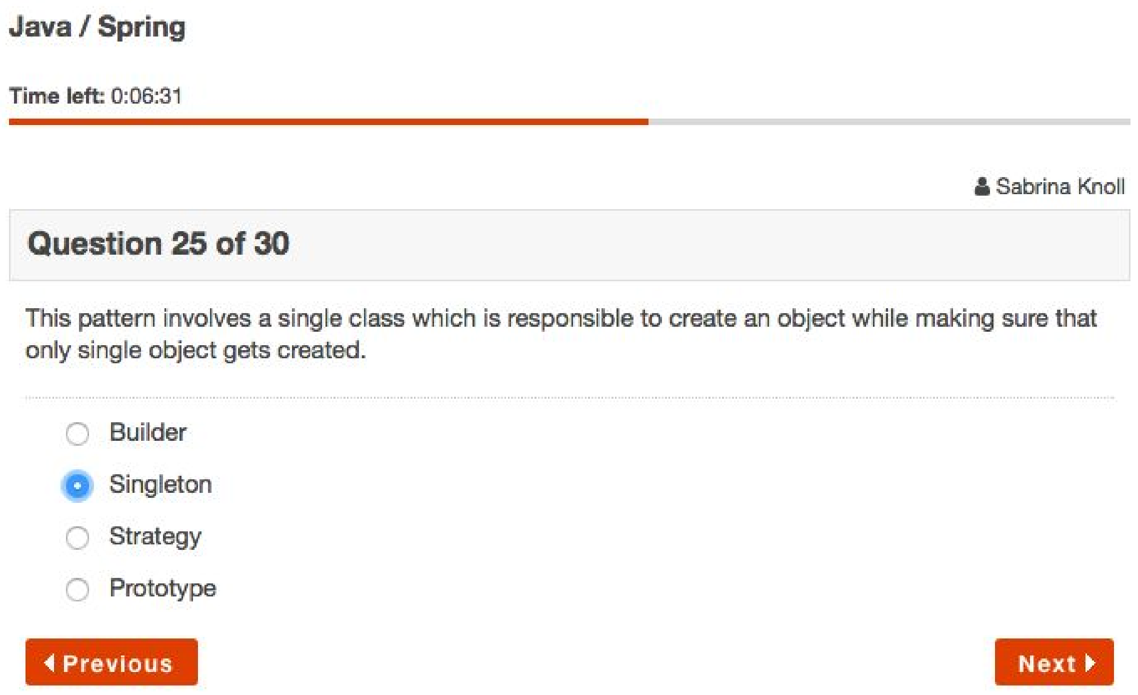
Time left: 0:06:41

Sabrina Knoll

Question 24 of 30

Which one of the following CRUD operations / HTTP method associations are valid ?

- [ ] **A)** Create - POST
- [ ] **B)** Create - PUT
- [ ] **C)** Read - PATCH
- [✓] **D)** Read - GET
- [✓] **E)** Update - POST
- [ ] **F)** Update - PUT
- [ ] **G)** Delete - DELETE
- [ ] **H)** Delete - REMOVE



Java / Spring

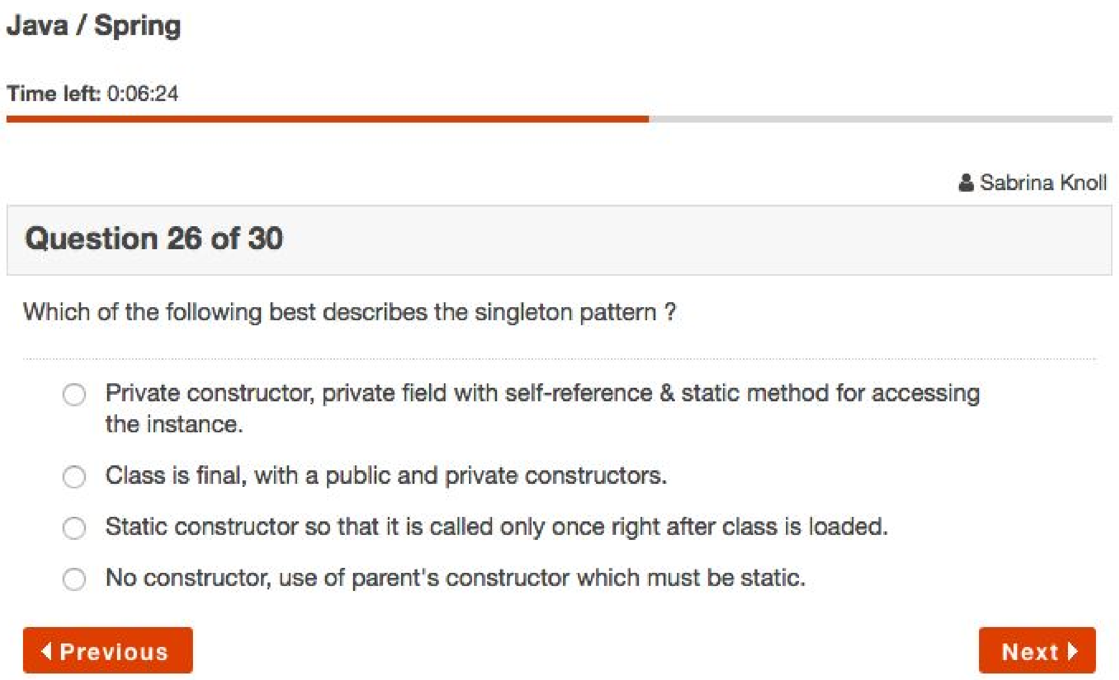
Time left: 0:06:31

Sabrina Knoll

Question 25 of 30

This pattern involves a single class which is responsible to create an object while making sure that only single object gets created.

- ○ Builder
- [✓ Singleton]
- ○ Strategy
- ○ Prototype



Java / Spring

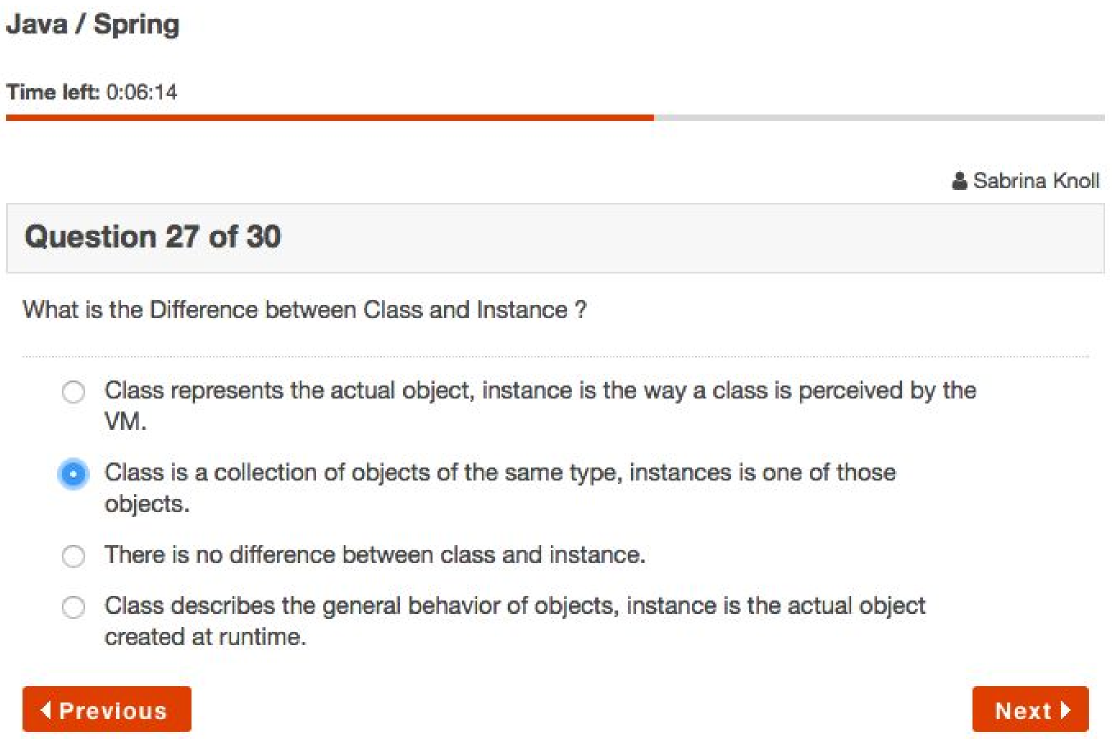
Time left: 0:06:24

Sabrina Knoll

Question 26 of 30

Which of the following best describes the singleton pattern ?

- Private constructor, private field with self-reference & static method for accessing the instance.
- Class is final, with a public and private constructors.
- Static constructor so that it is called only once right after class is loaded.
- No constructor, use of parent's constructor which must be static.



Java / Spring

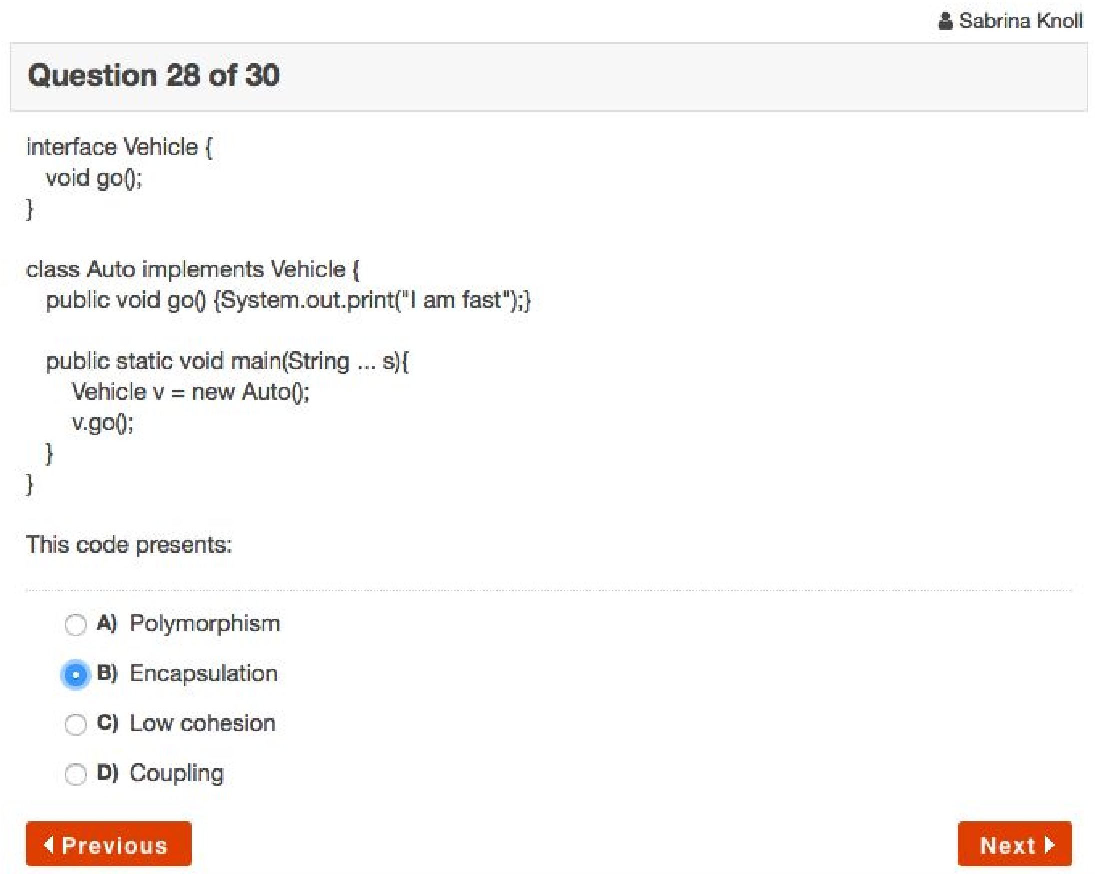
Time left: 0:06:14

Sabrina Knoll

Question 27 of 30

What is the Difference between Class and Instance ?

- Class represents the actual object, instance is the way a class is perceived by the VM.
- [✓ Class is a collection of objects of the same type, instances is one of those objects.]
- There is no difference between class and instance.
- Class describes the general behavior of objects, instance is the actual object created at runtime.



Sabrina Knoll

Question 28 of 30

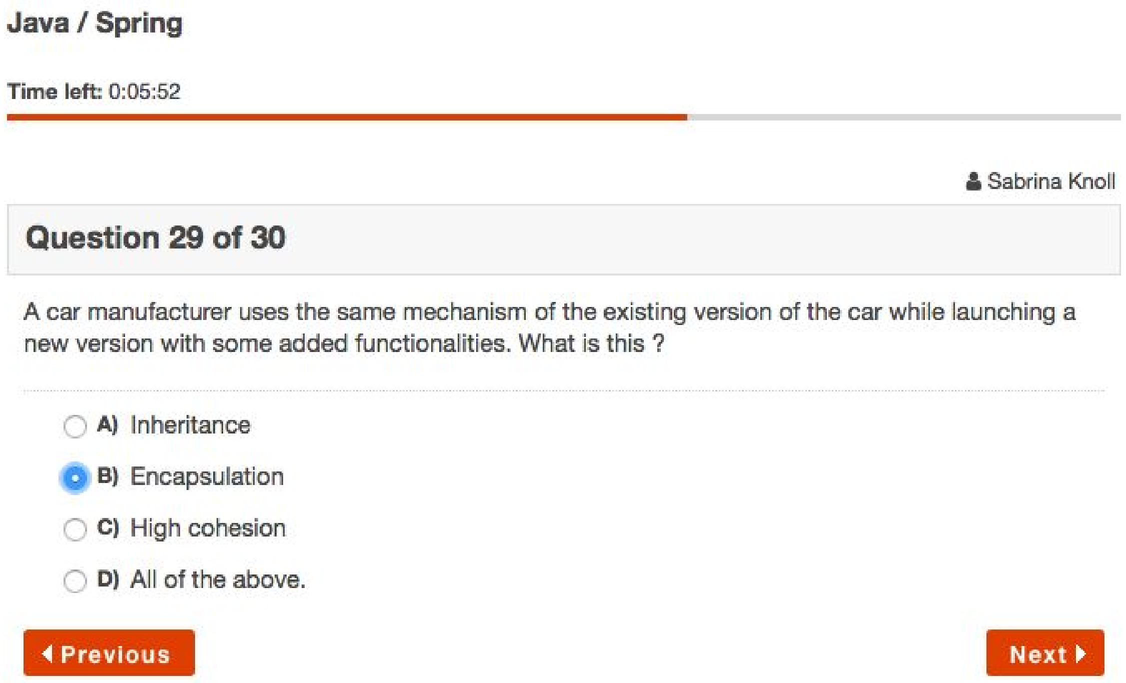
```
interface Vehicle {
  void go();
}

class Auto implements Vehicle {
  public void go() {System.out.print("I am fast");}

  public static void main(String ... s){
     Vehicle v = new Auto();
     v.go();
  }
}
```

This code presents:

- A) Polymorphism
- [✓ B) Encapsulation]
- C) Low cohesion
- D) Coupling



Java / Spring

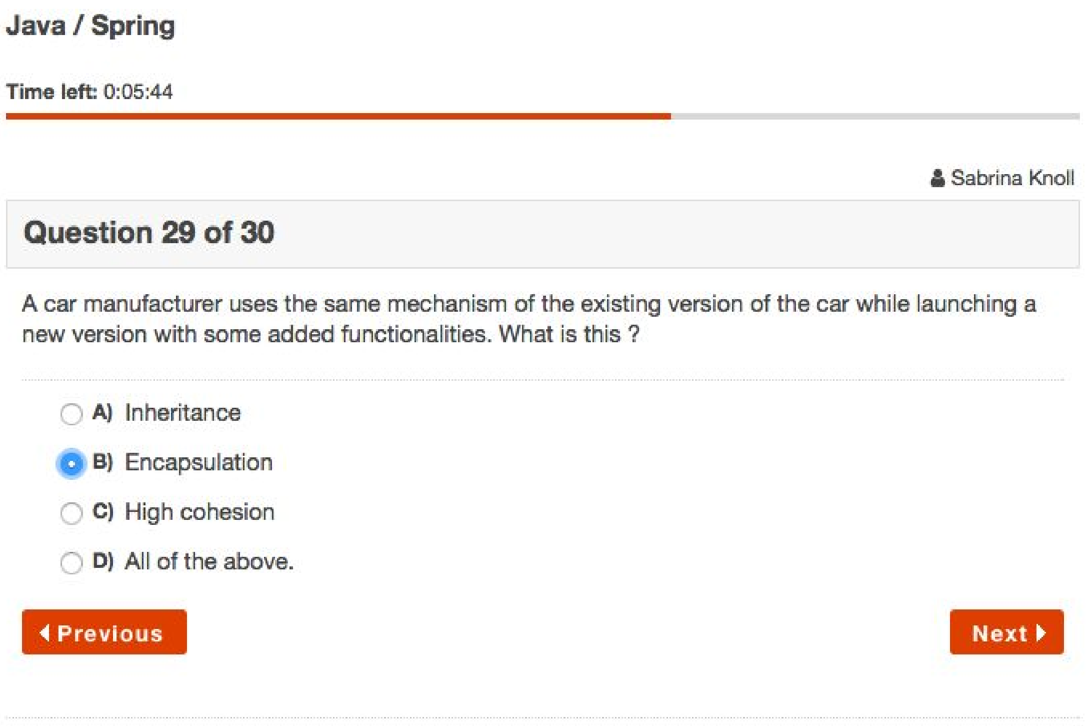
Time left: 0:05:52

Sabrina Knoll

Question 29 of 30

A car manufacturer uses the same mechanism of the existing version of the car while launching a new version with some added functionalities. What is this ?

- A) Inheritance
- [✓ B) Encapsulation]
- C) High cohesion
- D) All of the above.



Java / Spring

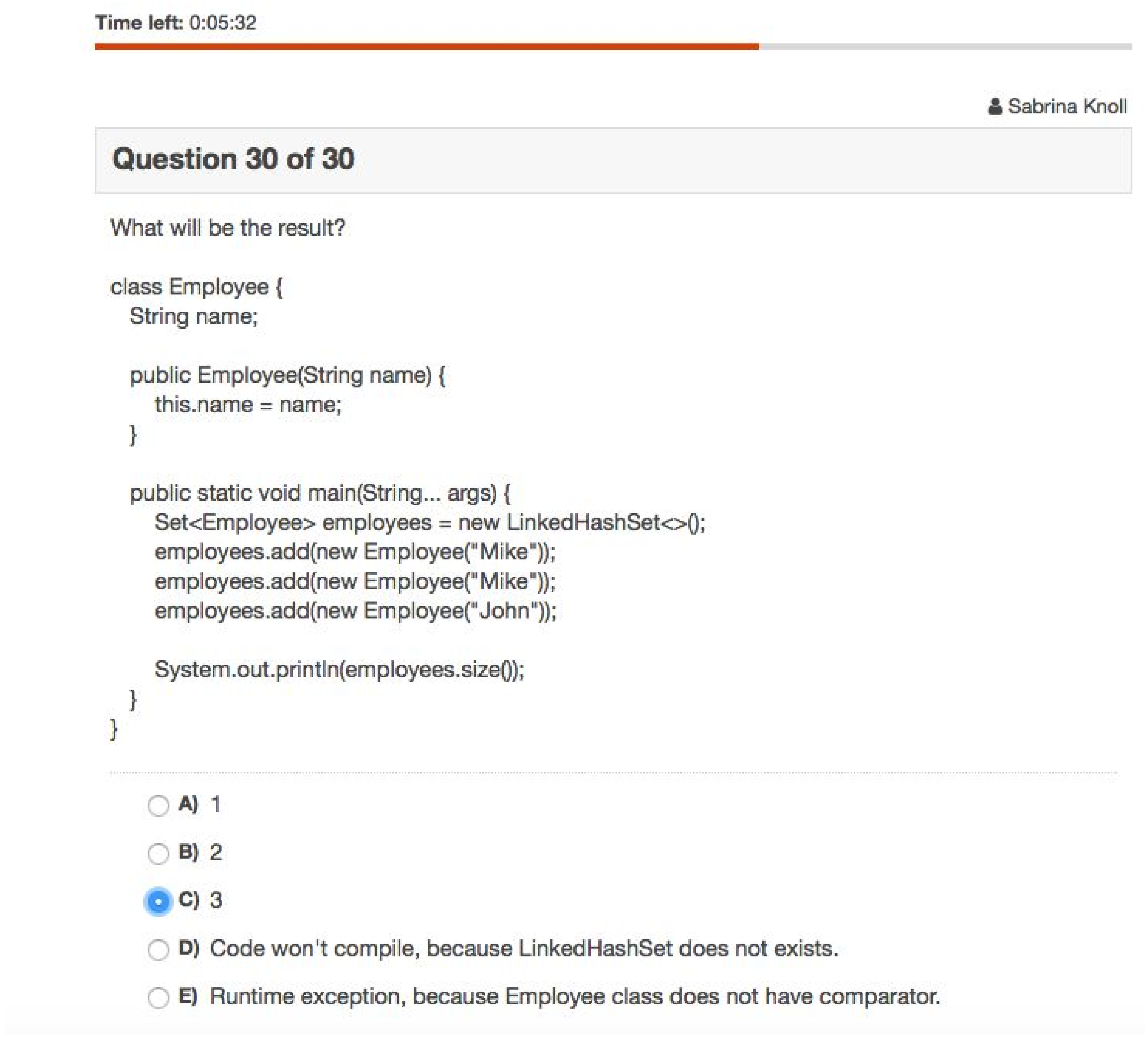
Time left: 0:05:44

Sabrina Knoll

Question 29 of 30

A car manufacturer uses the same mechanism of the existing version of the car while launching a new version with some added functionalities. What is this ?

- **A)** Inheritance
- [✓ **B)** Encapsulation]
- **C)** High cohesion
- **D)** All of the above.



Time left: 0:05:32

Sabrina Knoll

Question 30 of 30

What will be the result?

```
class Employee {
  String name;

  public Employee(String name) {
     this.name = name;
  }

  public static void main(String... args) {
     Set<Employee> employees = new LinkedHashSet<>();
     employees.add(new Employee("Mike"));
     employees.add(new Employee("Mike"));
     employees.add(new Employee("John"));

     System.out.println(employees.size());
  }
}
```

- **A)** 1
- **B)** 2
- [✓ **C)** 3]
- **D)** Code won't compile, because LinkedHashSet does not exists.
- **E)** Runtime exception, because Employee class does not have comparator.