1. What are the principles of encapsulation? (Select all that apply)

Show Results   18/19 Students Answered

- **A** Bundling object state and behavior ✓
- **B** Managing internal logic and consistency ✓
- **C** Validating user input ✗
- **D** Restricting direct access to internal properties ✓   fields private and methods public
- **E** Overloading methods ✗

2. Given the following code, what's the setter for lastname?

```
public class Person {
private String lastname;
...
}
```

String lastname
↓
public   void   setLastname( )

Show Results   17/19 Students Answered

A ✗
```
String setLastname(String lastname) {
this.lastname = lastname;
return lastname;
```

B ✗
```
void setLastname() {
lastname = lastname;
}
```

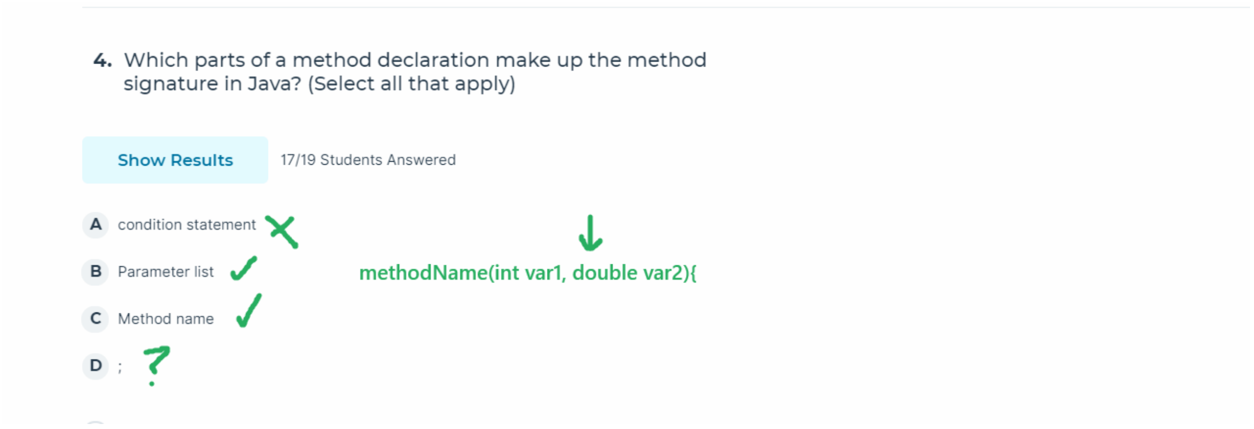
C ✗
```
void lastnameSet(String lastname) {
this.lastname = lastname;
}
```

D ←
```
void setLastname(String lastname) {
```

3. What does it mean to define a property or method as static?

Show Results   17/19 Students Answered

- **A** The property or method isn't available to external callers.   access modifier - private, public ✗
- **B** The property or method is encrypted. ✗
- **C** The property or method belongs to the class, and not an instance. ✓
- **D** The property or method contains noise and nothing useful. ✗ ?

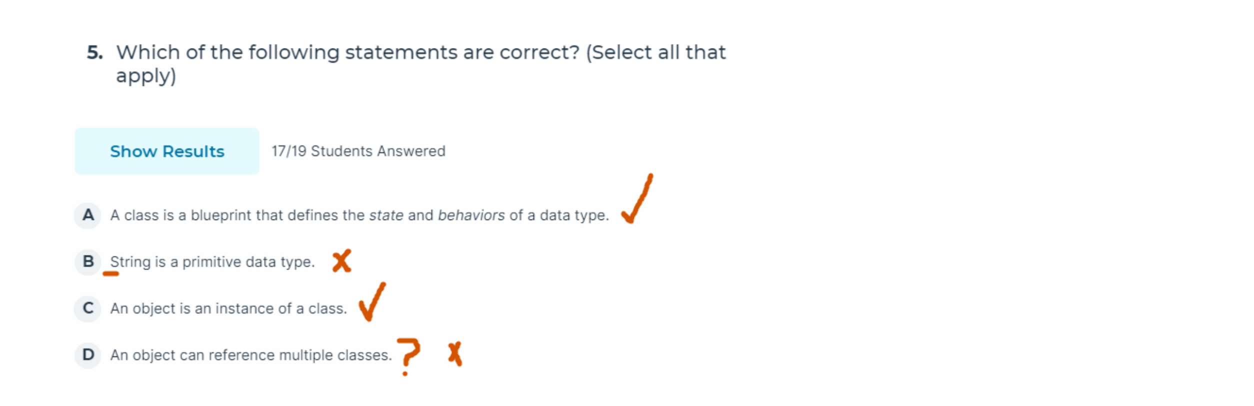


4. Which parts of a method declaration make up the method signature in Java? (Select all that apply)

Show Results 17/19 Students Answered

- A condition statement ✗
- B Parameter list ✓
- C Method name ✓
- D ; ?

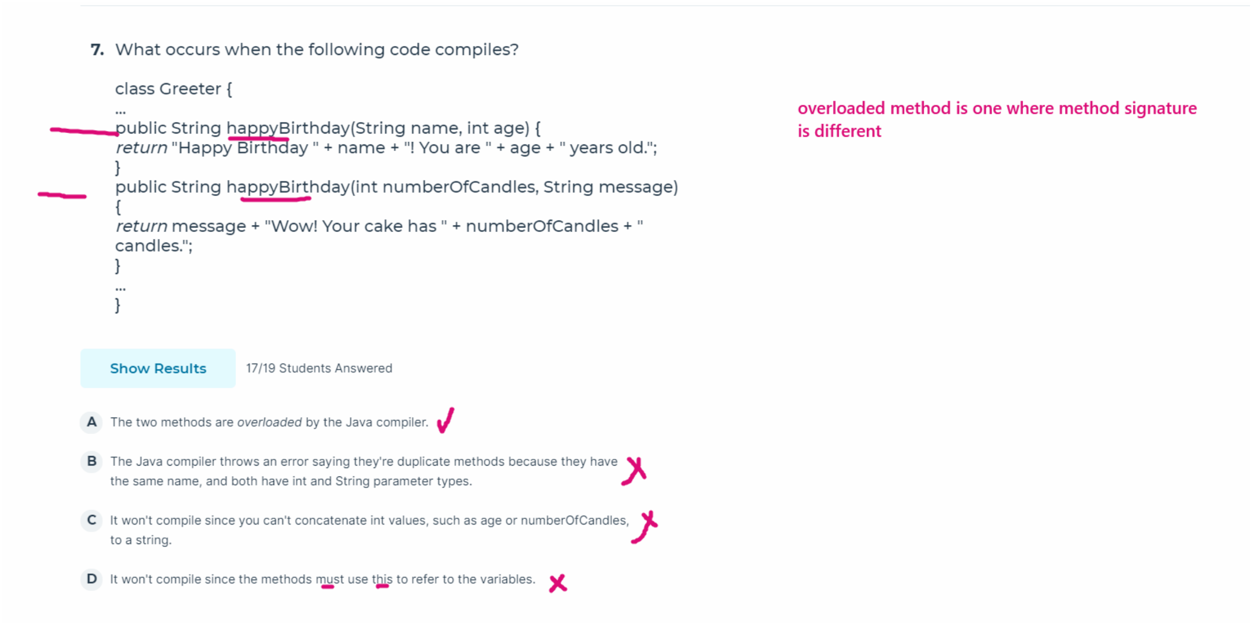
↓
methodName(int var1, double var2){



5. Which of the following statements are correct? (Select all that apply)

Show Results 17/19 Students Answered

- A A class is a blueprint that defines the *state* and *behaviors* of a data type. ✓
- B String is a primitive data type. ✗
- C An object is an instance of a class. ✓
- D An object can reference multiple classes. ? ✗



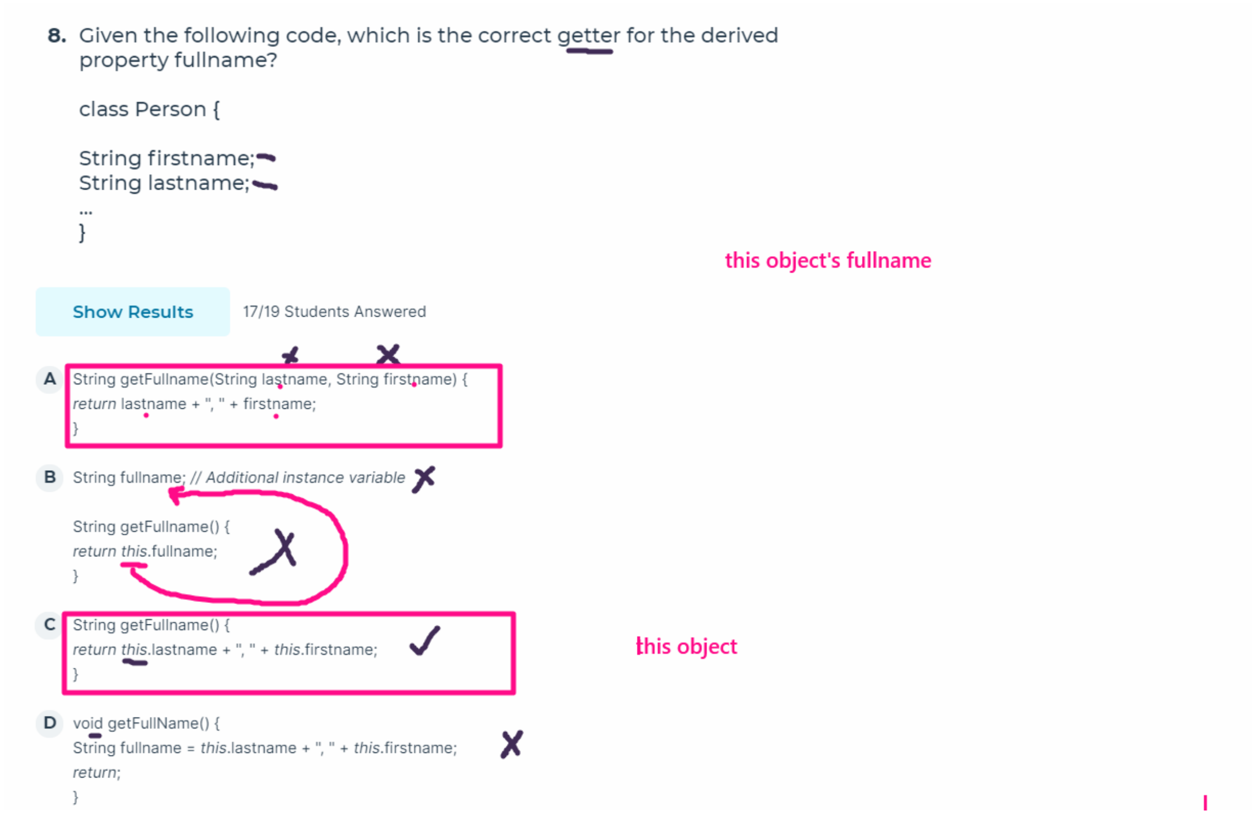
7. What occurs when the following code compiles?

```
class Greeter {
...
public String happyBirthday(String name, int age) {
return "Happy Birthday " + name + "! You are " + age + " years old.";
}
public String happyBirthday(int numberOfCandles, String message)
{
return message + "Wow! Your cake has " + numberOfCandles + "
candles.";
}
...
}
```

overloaded method is one where method signature is different

Show Results 17/19 Students Answered

- A The two methods are *overloaded* by the Java compiler. ✓
- B The Java compiler throws an error saying they're duplicate methods because they have the same name, and both have int and String parameter types. ✗
- C It won't compile since you can't concatenate int values, such as age or numberOfCandles, to a string. ✗
- D It won't compile since the methods must use this to refer to the variables. ✗



8. Given the following code, which is the correct getter for the derived property fullname?

```
class Person {

String firstname;
String lastname;
...
}
```

this object's fullname

Show Results 17/19 Students Answered

A ✗ ✗
```
String getFullname(String lastname, String firstname) {
return lastname + ", " + firstname;
}
```

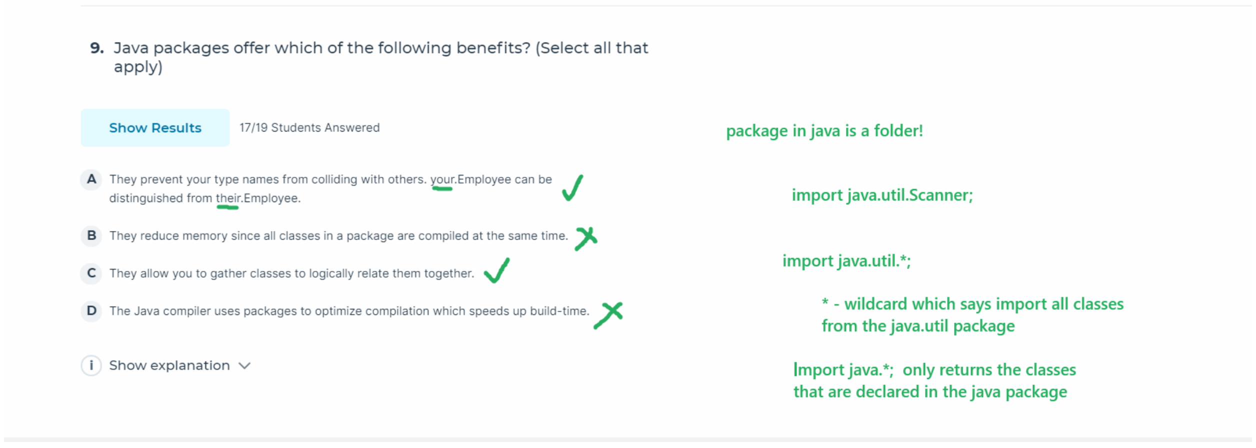
B ✗
```
String fullname; // Additional instance variable

String getFullname() {
return this.fullname;
}
```
✗

C ✓
```
String getFullname() {
return this.lastname + ", " + this.firstname;
}
```

this object

D ✗
```
void getFullName() {
String fullname = this.lastname + ", " + this.firstname;
return;
}
```



9. Java packages offer which of the following benefits? (Select all that apply)

Show Results 17/19 Students Answered

package in java is a folder!

A They prevent your type names from colliding with others. your.Employee can be distinguished from their.Employee. ✓

B They reduce memory since all classes in a package are compiled at the same time. ✗

C They allow you to gather classes to logically relate them together. ✓

D The Java compiler uses packages to optimize compilation which speeds up build-time. ✗

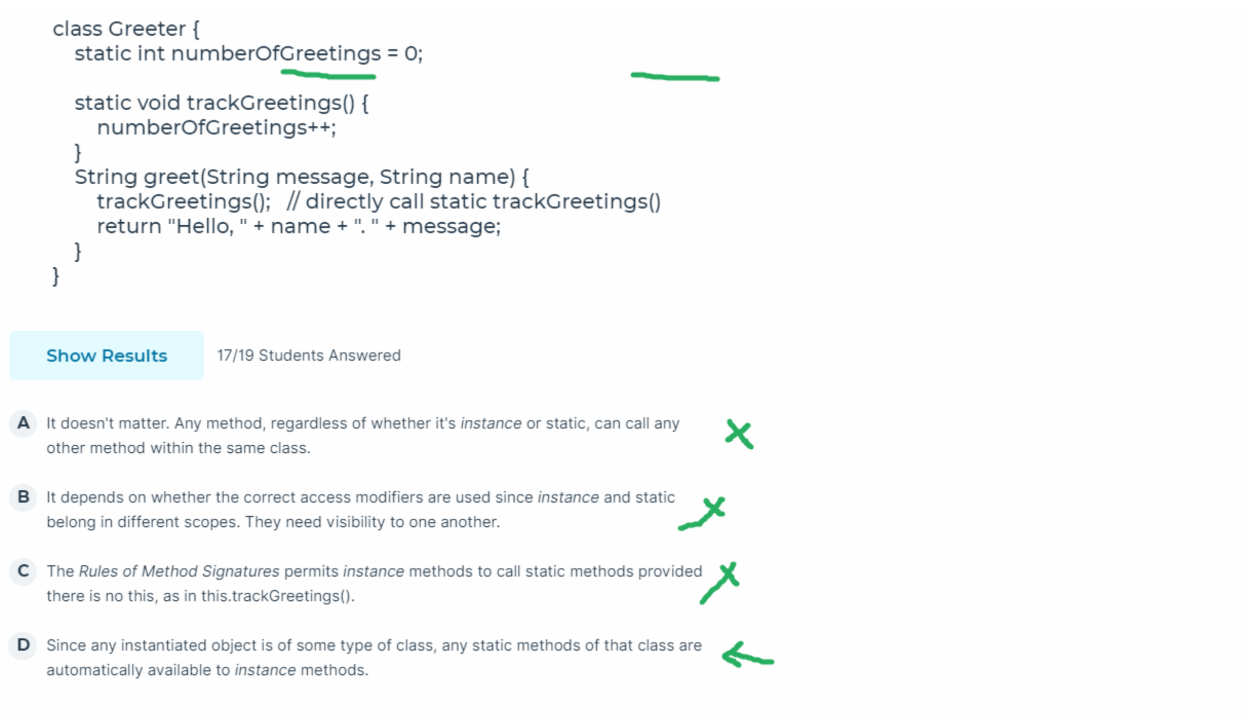
Show explanation

import java.util.Scanner;

import java.util.*;

* - wildcard which says import all classes from the java.util package

Import java.*; only returns the classes that are declared in the java package



```
class Greeter {
   static int numberOfGreetings = 0;

   static void trackGreetings() {
      numberOfGreetings++;
   }
   String greet(String message, String name) {
      trackGreetings();   // directly call static trackGreetings()
      return "Hello, " + name + ". " + message;
   }
}
```

Show Results   17/19 Students Answered

A It doesn't matter. Any method, regardless of whether it's *instance* or static, can call any other method within the same class. ✗

B It depends on whether the correct access modifiers are used since *instance* and static belong in different scopes. They need visibility to one another. ✗

C The *Rules of Method Signatures* permits *instance* methods to call static methods provided there is no this, as in this.trackGreetings(). ✗

[✓ **D** Since any instantiated object is of some type of class, any static methods of that class are automatically available to *instance* methods.] ←