socrative
A showbie PRODUCT

Answer Key
February 6, 2024

Loops, Collections, Encapsulation

1. A String is a Java Primitive

T True

✓ F False

i Primitive data types are managed on the Stack, and reference types are managed on the Heap.

Primitives:
byte, short, int, long, float, double, boolean, char

2.
```
int answer = 0;
for(int i = 0; i < 10; i++) {
   if(i % 2 == 0) {
      answer = answer + i;
   }
}

System.out.println(answer);
```

What value is displayed by this program?

20

i The for loop will loop 10 times with the following values:
0 1 2 3 4 5 6 7 8 9

The if statement will only include those numbers that have a remainder of 0 when divided by 2 (it will only add even numbers)
0 + 2 + 4 + 6 + 8 = 20

3.
```
int sum = 0;
for(int i = 16; i > 0; i -= 4) {
   sum += i;
}
System.out.println(sum);
```

What value does the preceding fragment display?

40

i The loop begins at 16 and counts down until it gets to 0, but it decrements by 3
16 12 8 4 0

The numbers are added
16 + 12 + 8 + 4 = 40

4.
```
1)  int sum = 0;
2)  for(int i = 0; i < 10; i++) {
3)      sum += i;
4)  }
5)
6)  int count = i;
```

Which line will causes an error?

A Line 1

B Line 2

C Line 3

D Line 4

✓ E Line 6

F There is no error

i Since the variable **i** is declared with the loop, it's scope is limited to the loop. You cannot access **i** outside of the loop's curly braces

5.
```
int[] prices = { 5, 3, 18, 4 };

prices[2] = 1;

for(int index = 0; index < prices.length; index++) {
    int price = prices[index]
    System.out.print(price + " ");
}
```

What will be displayed on the sreen?

5 3 1 4
5 3 1 4

i All arrays begin at index 0 (because the array gives access to the first number without needing to shift or offset)

prices[2] refers to the 3rd number in the array { 5, 3, **18**, 4 }

The line of code immediately prior to the loop results in { 5, 3, **1**, 4 }

Inside the loop the System.out.**print** prints the number and an empty space " ", but no new line... so the output is:
5 3 1 4

All on a single line

6. **Given the following for loop:**

```
for(int i = 0; i < 5; i++) {
    int sum = 0;
    sum += i;
    System.out.println(sum + " ");
}
```

What is the printed output of the preceding code?

0 1 2 3 4
0 1 2 3 4

i Since the variable sum is declared **inside** the for loop, **sum** will always be reset to **0.**

Then we add the value of i to 0, which will always be the same value.

This is a common mistake when developing. The variable **sum**, needs to be declared outside the for loop in order to be able to remember and add to the sum with each iteration of the loop.

7.
```
int[] numbers = new int[10];
for(int index = 0; index < numbers.length; index++) {
   numbers[index] = index * 3;
}

int result = numbers[3];
```

What is the value of result after this code executes?

9

i The loop executes 10 times, with the value of index changing to ...
0 1 2 3 4 5 6 7 8 9

with each iteration the "numbers" array is filled with the value of (index * 3)

numbers[0] = (0 * 3) → 0
numbers[1] = (1 * 3) → 3
numbers[2] = (2 * 3) → 6
numbers[3] = (3 * 3) → 9
numbers[4] = (4 * 3) → 12
etc

8. Given the List declaration below, how do you access the *third* item?

```
ArrayList<String> animals = new ArrayList<>();
animals.add("Dog");
animals.add("Cat");
animals.add("FIsh");
animals.add("Bear");
```

✓ **A** animals.get(2)

B animals.get(3)

C animals.get("Fish")

D animals[2]

ℹ **Correct: A**
Correct. Lists are zero-based indexes like arrays, so you use the .get() method.

Wrong:
B - This retrieves the fourth item because Lists are zero-based indexes like arrays.
C - You can't retrieve a List item by its value.
D - This isn't valid syntax to retrieve a value from a List. This only works on arrays.

9. What does the <T> indicate when used like **ArrayList<T>**?

A It means that it's a collection.

B The List can only hold a datatype called T.

✓ **C** You need to specify a datatype in place of T.

D Nothing. It's a typo.

ℹ **Correct: C**
The T is a placeholder for a datatype to be specified by the developer. The datatype goes in the angle brackets <>.

Wrong:
A - The package of the List class indicates that it's a collection.
B - That could be possible, but a developer probably wouldn't use T as it isn't a very descriptive name.
D - It's not a typo.

10. Select the ways an ArrayList differs from an array (select all that apply):

✓ **A** An ArrayList's length doesn't need to be declared.

B You can add different data types to an ArrayList.

✓ **C** Elements can be easily added to and removed from the middle of an ArrayList.

D You can't access an ArrayList element at a particular index.

✓ **E** You use a different method to retrieve the number of elements in an ArrayList.

ℹ **Correct:**
A - An ArraList grows and shrinks as it's needed, unlike an array that has a fixed length.
C - ArrayLists have methods that you can use to add and remove elements from the middle of an ArrayList, unlike arrays.

E - ArrayLists have a different method for retrieving the number of elements.

Wrong:
B - Like arrays, Lists are bound to the data type it was declared with.
D - You can access an ArrayList element at a particular index.

11. What method do you use to get the number of elements in an ArrayList?

A .length()

B .amount()

✓ C .size()

D .count()

i **Correct: C**
This is the correct method.

Wrong:
A - length is the *attribute* used for arrays.
B,D - These aren't valid methods.

12. Given an ArrayList<String>, what does the following line of code do?

animals.add(3, "Ice Cream");

A Replaces the fourth item in the list with the text "Ice Cream".

B Adds "Ice Cream" as the third item in the list and shifts everything else down

C Replaces the third item in the list with the text "Ice Cream".

✓ D Adds "Ice Cream" as the fourth item in the list and shifts everything else down

i **Correct: D**
This is correct. The 3 indicates the zero-based index of where to insert the item. The add() method adds a new item to the list, it does not replace.

Wrong:
A - The add() method adds a new item to the list, it does not replace.
B - The 3 indicates the zero-based index of where to insert the item, so the correct insertion point is 4.
C - The 3 indicates the zero-based index of where to insert the item, so the correct insertion point is 4. The add() method adds a new item to the list, it does not replace.

13. What is Encapsulation? (select all that apply)

✓ A Information hiding

B Creating global variables

✓ C Implementation hiding

D Data validation

i **Correct: A, C**
Encapsulation is the process of grouping related data and methods together into a class. The class is responsible for hiding the data (information).

Classes also have methods. Public methods are exposed, but the implementation of those methods is maintained within the class as well.

By grouping all related data and methods within a class we ensure that the data and methods are highly cohesive.

14.

Given the class definition for a Hotel in the image, what is the appropriate setter for the hotel name?

A public String setName(String name) {
this.name = name;
return name;
}

B public void setName() {
name = name;
}

C public void name(String name) {
this.name= name;
}

✓ **D** public void setName(String name) {
this.name= name;
}

Correct: D

The setter of a variable should have the following characteristics
- it is public
- returns nothing (void)
- begins with the word **set**
- takes a single input parameter that is the same datatype as the value being set
- updates / modifies the value of the private variable

The signature is:
public void setName(String name)

15.

```
2 public class Hotel
3 {
4     private String name;
5
6 }
```

Given the class definition for a Hotel in the image, what is the appropriate getterfor the hotel name?

A
```
public String getName(String name) {
    return name;
}
```

B
```
public void getName() {
    return name;
}
```

✓ C
```
public String getName() {
    return name;
}
```

D
```
public String name() {
    return name;
}
```

i **Correct: C**

The getter of a variable should have the following characteristics
- it is public
- returns the same datatype as the variable
- begins with the word **get**
- accepts NO input parameters
- returns the current value of the private variable

The signature is:
public String getName()

16. Which of the following functions has the same signature as the findCustomer?

```
public Customer findCustomer(String firstName, String lastName)
{...}
```

A
```
public Customer searchCustomer(String firstName, String lastName)
{...}
```

B
```
public Customer findCustomer(int customerId)
{...}
```

✓ C
```
public Customer findCustomer(String city, String state)
{...}
```

D
```
private Customer findCustomer(String firstName, String middleName, String lastName)
{...}
```

i **Correct: C**

The signature of a method is important because Java supports overloading methods. When a method is called, Java needs to determine which code to execute. Since method names can be duplicated, the entire signature must be used to make that determination.

Signatures must be unique - in other words, you are not allowed to have 2 methods with the same signature within a class.

The signature of a method consists of:
- the method name
- the number of input parameters
- the data type of the input parameters

What is NOT included as part of the signature:
- the method return type
- the visibility/accessibility of the method
- the names of the input parameters

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

✓ **A** A class is a template that defines the variables and methods of a data type.

B A class can be overloaded.

C A class defines a primitive data type.

✓ **D** An object is an instance of a class.

Correct: A, D

Classes are like blueprints, or templates that define what capabilities an object will have when it is created.

An object is created from the class definition by creating an instance.

Incorrect:
Classes cannot be overloaded. Overloading is a term used to define when two or more methods are created, that share the same method name.

An object is managed on the heap, which means that it is not a primitive type.

18. What access modifier hides properties and methods?

A static

✓ **B** private

C final

D public

Correct: B

Variables or methods can be hidden by setting them as private.

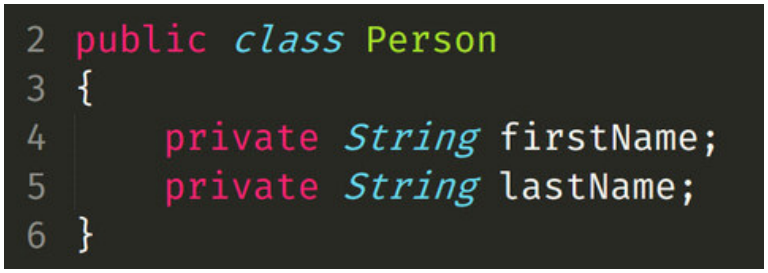
Incorrect:
Static is a modifier that is used to mark a variable as shared. Static variables are managed by the class and not by each instance of the class.

Final - in variable definitions
final is a modifer that is used to create constants.

Final - in class/method definitions
final is a modifier that ensures a class/method cannot be extended

public is a modifier to make a variable, class or method visible to code outside of the class.

19.



```
2 public class Person
3 {
4     private String firstName;
5     private String lastName;
6 }
```

Given the following code, which is the correct getter for the derived property fullName?

A public String getFullName(String lastName, String firstName){
    return lastName + ", " + firstName;
}

B private String fullName;   //Additional instance variable

public String getFullName(){
    return this.fullName;
}

✓ **C** public String getFullName(){
    return this.lastName + ", " this.firstName;
}

D public void getFullName(){
    String fullName = this.lastName + ", " + this.firstName;
}

i **Correct: C**

getFullName() is a derived property. This means that there is not direct backing variable (fullName) that stores the value. getFullName() uses 2 other variables (firstName and lastName) to calculate and return the full name value.

Derived properties may also be referred to as calculated properties or calculated getters. They are public and should always begin with the word **get**.

20. What benefits do Java packages provide? (Select all that apply)

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

✓ B They prevent your class names from colliding with others.
java.util.ArrayList can be distinguished from
org.mycompany.ArrayList

✓ C They allow you to organize classes into folders.

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

i **Correct: B, C**

The purpose of packages is strictly used for code organization. Like a file system, java packages employ a folder structure to organize your classes.

File names (and therefore class names) must be unique within any folder. By creating a folder/package structure you can reuse class names that exist in a different package. This helps avoid class name conflicts.

Incorrect:
Packages do not have any effect on the compiler optimization or size of the compiled code.