04/26/2017

# Java 111 Chapter 9, 17 Quiz

Total Questions: 23

Most Correct Answers: **#11**

Least Correct Answers: **#20**

## 1. Instance variables are variables declared inside a method or method parameter.

1/12 A True

1/12 B True only in an abstract class

9/12 [✓ C] False

## 2. Which of the following is are true?

0/12 A If an object reference is declared as a local variable it goes on the heap.

8/12 [✓ B] Local variables live on the stack in the frame corresponding to the method where the variables are declared.

7/12 [✓ C] All objects live in the heap regardless of whether the reference is a local or instance variable.

11/12 [✓ D] Instance variables are variables declared inside a class but outside any method.

## 3. In the code example below, which is the object reference variable?

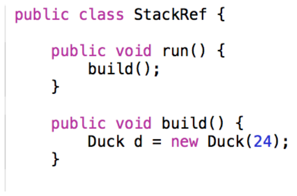
0/12 A Duck()

0/12 B 24

11/12 [✓ C] d

0/12 D new Duck()

0/12 E Duck



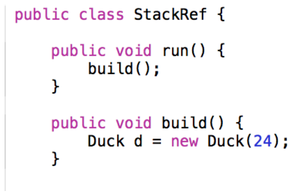
```
public class StackRef {

    public void run() {
        build();
    }

    public void build() {
        Duck d = new Duck(24);
    }
```

## 4. In the code example below, where will the Duck object live?

10/12 [✓ A] Heap

1/12 B Stack

0/12 C Frame



```
public class StackRef {

    public void run() {
        build();
    }

    public void build() {
        Duck d = new Duck(24);
    }
```

## 5. The currently executing method is located where in memory?

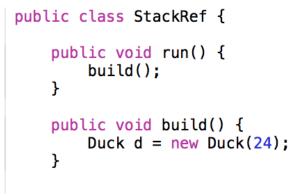
- 0/12 (A) Bottom of the heap
- 0/12 (B) Top of the heap
- 0/12 (C) Bottom of the stack
- 11/12 [✓ D] Top of the stack

## 6. When is an object eligible for garbage collection?

- 7/12 [✓ A] When the reference is assigned to another object
- 7/12 [✓ B] When the reference is set to null
- 1/12 (C) As soon as it is instantiated
- 11/12 [✓ D] When the reference variable goes out of scope (it is no longer pointing to the object)

## 7. In the code example below, how long will "d" live on the stack?

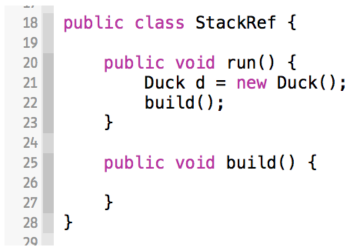
- 2/12 (A) Until run() pops off the stack
- 5/12 [✓ B] Until build() pops off the stack
- 0/12 (C) Until the jvm is restarted
- 4/12 (D) d will live in the heap, not the stack



```java
public class StackRef {

    public void run() {
        build();
    }

    public void build() {
        Duck d = new Duck(24);
    }
```

## 8. In the code example below, how long will "d" live on the stack?

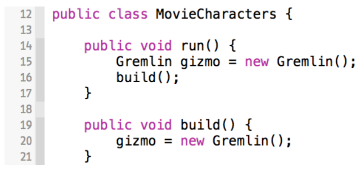
- 10/12 [✓ A] Until run() pops off the stack
- 0/12 (B) Until build() is added to the top of the stack
- 1/12 (C) Until build() pops off the stack
- 0/12 (D) Until d is garbage collected



```java
public class StackRef {

    public void run() {
        Duck d = new Duck();
        build();
    }

    public void build() {

    }
}
```

## 9. What, if anything, is wrong with this code snippet?

- 2/12 (A) The constructor for Gremlin is missing
- 8/12 [✓ B] gizmo is "scoped" only to the run method, so it can't be used anywhere else
- 0/12 (C) Nothing is wrong with this code
- 0/12 (D) The run() method should not have a return type of void
- 1/12 (E) When the build() method runs, gizmo from line 15 will have been garbage collected



```java
12  public class MovieCharacters {
13
14      public void run() {
15          Gremlin gizmo = new Gremlin();
16          build();
17      }
18
19      public void build() {
20          gizmo = new Gremlin();
21      }
```

## 10. Which of the following are true?

11/12 [✓ A] A constructor is the code that runs when somebody says "new" on a class type, like this: Duck d = new Duck();

2/12 B A constructor must have the same name as the class and a return type of New

10/12 [✓ C] If you do not put a constructor in your class, the compiler creates a default constructor

10/12 [✓ D] The default constructor created by the compiler has no arguments.

## 11. You can have more than one constructor in your class as long as the argument lists are different. This means you have overloaded constructors.

12/12 [✓ A] True

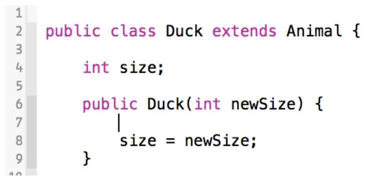
0/12 B False

## 12. All constructors in an object's inheritance tree must run when you make a new object.

6/12 [✓ A] True

6/12 B False

## 13. Type the code that should be entered on line 7 to call the Duck's super constructor.



```
1
2  public class Duck extends Animal {
3
4      int size;
5
6      public Duck(int newSize) {
7          |
8          size = newSize;
9      }
```

Anon anon01096363e2694f01

✓ super();

Anon anon2fc5d1dcd69f4841

✗ super.Duck(newSize);

Anon anon4638f6f81edf44da

✗ super.Animal();

Anon anon4ef611c3ff284c60

✗ Duck newDuck = new super();

Anon anon597107e212894f2e

✓ super();

Anon anon598590c81dac41dd

✗ super.Animal();

Anon anon5c063a97a713422d

```
public class UseADuck {

public static void main(String[] args) {
Duck duck = new Duck();
}
}
```

✗

**Anon anon5c67912b444a4b21**

✗ super.Animal()

**Anon anon9f3ef858446e4235**

✓ super();

**Anon anond42fa221e38a4cd4**

✓ super();

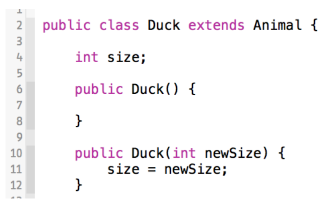
**Anon anonde35b2f741114219**

✗ this();

**Anon anone7bb968ca6034f61**

✗ this();

14. Type the code that should appear on line 7 to call the Duck's single arg constructor with a parameter of 14.



```
 2  public class Duck extends Animal {
 3
 4      int size;
 5
 6      public Duck() {
 7
 8      }
 9
10      public Duck(int newSize) {
11          size = newSize;
12      }
```

Anon anon01096363e2694f01

✗ Duck myDuck = new Duck(12);

Anon anon2fc5d1dcd69f4841

✓ this(14);

Anon anon4638f6f81edf44da

✓ this(14);

Anon anon4ef611c3ff284c60

✗ Duck newDuck = new Duck(14);

Anon anon597107e212894f2e

✓ this(14);

Anon anon598590c81dac41dd

✗ size(14);

Anon anon5c67912b444a4b21

✗ Duck one = new Duck(14);

Anon anon9f3ef858446e4235

✗ Duck(14);

Anon anond42fa221e38a4cd4

✗ new Duck(14);

Anon anonde35b2f741114219

✗ Duck bigDuck = Duck(14);

Anon anone7bb968ca6034f61

✗ this.size(14);

15. A constructor can have a call to super() OR this(), but NEVER both.

2/12 [✓ A] True

10/12 B False

16. It is good practice to keep source code and compiled code (class files) separate, but there is no way to do this in Java.

0/12 (A) True

12/12 [✓ (B) False]

17. One key feature of using packages is to prevent class name conflicts.

10/12 [✓ (A) True]

2/12 (B) False

18. You have a class called Project1 in this directory structure: src/edu/madisoncollege/javaprojects. Write the package statement that should appear at the top of the Project1 class.

**Anon anon01096363e2694f01**

✗ package Project1/javaprojects;

**Anon anon2fc5d1dcd69f4841**

✗ package java111.Projects.project5

**Anon anon4638f6f81edf44da**

✗ package src.edu.madisoncollege.javaprojects;

**Anon anon4ef611c3ff284c60**

✗ edu.madisoncollege.javaprojects

**Anon anon597107e212894f2e**

✗ package src.edu.madisoncollege.javaprojects;

**Anon anon598590c81dac41dd**

✗ package madisoncollege.javaprojects;

**Anon anon5c063a97a713422d**

✗ package com.edu.madisoncollege.javaprojects

**Anon anon5c67912b444a4b21**

✓ package edu.madisoncollege.javaprojects;

**Anon anon9f3ef858446e4235**

✗ package src.edu.madisoncollege.javaprojects.Project1;

**Anon anond42fa221e38a4cd4**

✓ package edu.madisoncollege.javaprojects;

**Anon anonde35b2f741114219**

✗ package edu.madisoncollege.javaprojects

**Anon anone7bb968ca6034f61**

✓ package edu.madisoncollege.javaprojects;

**19. My class Book that has a package structure of java111.project5.labs and "lives" in the projects/src/java111/project5/labs directory. What is the proper way to compile Book into its proper package structure in the classes directory (assume projects/classes/java111/project5/labs)?**

0/12 (A) cd to the labs directory, then type: javac Book.java

8/12 [✓ B] cd to the projects directory, then type: javac -classpath classes -d classes java111/project5/labs/Book.java

0/12 (C) cd to the projects directory, then type: javac java111/project5/labs5/Book.java

4/12 (D) cd to the projects directory, then type: javac -classpath classes -d classes java111.project5.labs.Book.java

**20. Given, "javac -classpath classes -d classes java111/project5/labs/Book.java", what does the -d parameter do?**

1/12 (A) Tells the compiler to debug the Book class

7/12 [✓ B] Tells the compiler to build the directories java111, project5, labs in the proper structure if they do not already exist

5/12 [✓ C] Tells the compiler to send the compiled classes to classes/java111/project5/labs/

2/12 (D) Tells the compiler run javadoc on the Book class

**21. How can I run my Book class which resides in projects/classes/java111/project5/labs/Book.class**

0/12 (A) cd to the src directory and type: java Book

0/12 (B) cd to the classes directory and type: java Book

2/12 (C) cd to the projects directory and type: java java111.project5.labs.Book

10/12 [✓ D] cd to the projects directory and type: java -classpath classes java111.project5.labs.Book

0/12 (E) cd to the projects directory and type: java -classpath classes Book

**22. Which of the following are valid javadoc comments?**

1/12 (A) // @author aSchmidt

1/12 (B) //* @author aSchmidt *//

10/12 [✓ C] /** @author aSchmidt */

0/12 (D) /* @author aSchmidt */

23. **Javadoc can be created for all classes in my java111.project5 package using what command from my projects directory?**

| | | |
|---|---|---|
| 3/12 | [✓ A] | javadoc -d docs -sourcepath src java111.project5 |
| 1/12 | B | javadoc -d docs -sourcepath src/java111.* |
| 5/12 | C | javadoc -d docs -sourcepath src java111/project5 |
| 3/12 | D | javadoc -d docs java111.project5.* |
| 0/12 | E | It's not possible, javadoc can only be run for one class at a time |