socrative

Java 111 Chapter 9, 17 Quiz

24 Questions

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

1/1 **A** True

0/1 **B** True only in an abstract class

0/1 **C** False

2. Which of the following is are true?

1/1 **A** If an object reference is declared as a local variable it goes on the heap.

1/1 **B** Local variables live on the stack in the frame corresponding to the method where the variables are declared.

1/1 **C** All objects live in the heap regardless of whether the reference is a local or instance variable.

0/1 **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?

```
public class StackRef {

    public void run() {
        build();
    }

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

0/1 **A** Duck()

0/1 **B** 24

1/1 **C** d

0/1 **D** new Duck()

0/1 **E** Duck

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

```
public class StackRef {

    public void run() {
        build();
    }

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

0/1 **A** Heap

1/1 **B** Stack

0/1 **C** Frame

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

0/1 **A** Bottom of the heap

1/1 **B** Top of the heap

0/1 **C** Bottom of the stack

0/1 **D** Top of the stack

6. When is an object eligible for garbage collection?

0/1 A When the reference is assigned to another object

0/1 B When the reference is set to null

0/1 C As soon as it is instantiated

1/1 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?

```
public class StackRef {

    public void run() {
        build();
    }

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

0/1 A Until run() pops off the stack

1/1 B Until build() pops off the stack

0/1 C Until the jvm is restarted

0/1 D d will live in the heap, not the stack

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

```
18 public class StackRef {
19
20     public void run() {
21         Duck d = new Duck();
22         build();
23     }
24
25     public void build() {
26
27     }
28 }
```

1/1 A Until run() pops off the stack

0/1 B Until build() is added to the top of the stack

0/1 C Until build() pops off the stack

0/1 D Until d is garbage collected

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

```
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     }
```

0/1 A The constructor for Gremlin is missing

0/1 B gizmo is "scoped" only to the run method, so it can't be used anywhere else

0/1 C Nothing is wrong with this code

1/1 D The run() method should not have a return type of void

0/1 E When the build() method runs, gizmo from line 15 will have been garbage collected

10. Which of the following are true?

1/1 A A constructor is the code that runs when somebody says "new" on a class type, like this: Duck d = new Duck();

1/1 B A constructor must have the same name as the class and a return type of New

1/1 C If you do not put a constructor in your class, the compiler creates a default constructor

1/1 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.

1/1 A True

0/1 B False

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

1/1 A True

0/1 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 anon970f438f6aef4ebf**

1/1 Duck d = new Duck();

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

```
1
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 anon970f438f6aef4ebf**

1/1 newSize = 14;

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

1/1 A True

0/1 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.

0/1 A True

1/1 B False

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

1/1 A True

0/1 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 anon970f438f6aef4ebf**

1/1 package edu.madisoncollege.javaprojects.Project1;

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)?

0/1 **A** cd to the labs directory, then type: javac Book.java

0/1 **B** cd to the projects directory, then type: javac -classpath classes -d classes java111/project5/labs/Book.java

1/1 **C** cd to the projects directory, then type: javac java111/project5/labs5/Book.java

0/1 **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?

0/1 **A** Tells the compiler to debug the Book class

1/1 **B** Tells the compiler to build the directories java111, project5, labs in the proper structure if they do not already exist

0/1 **C** Tells the compiler to send the compiled classes to classes/java111/project5/labs/

0/1 **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

0/1 **A** cd to the src directory and type: java Book

0/1 **B** cd to the classes directory and type: java Book

1/1 **C** cd to the projects directory and type: java java111.project5.labs.Book

0/1 **D** cd to the projects directory and type: java -classpath classes java111.project5.labs.Book

0/1 **E** cd to the projects directory and type: java -classpath classes Book

22. Which of the following are valid javadoc comments?

0/1 **A** // @author aSchmidt

0/1 **B** //* @author aSchmidt *//

1/1 **C** /** @author aSchmidt */

0/1 **D** /* @author aSchmidt */

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

0/1 A javadoc -d docs -sourcepath src java111.project5

0/1 B javadoc -d docs -sourcepath src/java111.*

0/1 C javadoc -d docs -sourcepath src java111/project5

1/1 D javadoc -d docs java111.project5.*

0/1 E It's not possible, javadoc can only be run for one class at a time

24. Importing other packages and classes into your class is necessary:

0/1 A so that the compiler can find the .class files of all classes referenced in your class.

1/1 B so that the compiler can find the source code files of all classes referenced in your class.

0/1 C so that the compiler can put your new .class file in the correct location in your classes directory.

0/1 D so that the compiler knows where to find the source code of your class