|  |  |  |
| --- | --- | --- |
| **APCS Exposure Java** | **Exercises 09.01-07** | **Date:** |
| **Name:** | | **Period:** |

1. What is *encapsulation*?

2. Why is *OOP* so popular?

3. What is *inheritance*?

4. In inheritance, the existing class is called the \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ and the new class, which inherits the features of the existing class, is called the \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

5. In computer science an *is-a* relationship involves \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

6. In computer science a *has-a* relationship involves \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

7. Give an example of an *is-a* relationship.

8. Give an example of a *has-a* relationship.

Questions 9 through 11 involve project Java0902

9. Why do the 2 **Spider** objects look and behave like **Actor** objects?

10. In the heading: **class Spider extends Actor** which is *subclass* and which is the *superclass*?

11. The 2 **Spider** objects are created without names. What type of objects are they?

12. Why do we use *inheritance*?

13. How will a subclass behave is it has a method body that is completely empty?

14. Suppose you create a new **Frog** class in GridWorld that extends **Actor**. If you do not supply some type of image file for this file, what will the *frogs* look like when the project is executed?

15. Refer to the previous question. Suppose you have an image of a frog stored in a **gif** file.

What does this file have to be named if you want your **Frog** objects to resemble the image.

Questions 16 through 17 involve project Java0904

16. Is the **Spider** class a subclass, superclass, or both?

17. The **LazySpider** class redefined the **act** method with an empty method body.

Does this accomplish anything?

18. Suppose you have a class that is tested and works perfectly. You now need to create a enhanced version of this class for a specific purpose. Which if these 2 solutions is better:

a) Alter the existing class to meet the specific needs of the new class

b) Leave the existing class alone. Create a subclass that extends the existing class

and then add or redefine what ever methods are necessary.

Questions 19 through 22 involve program Java0901.java

19. Which keyword in the heading of the **Student** class is used to specify that it inherits from the **Person** class?

20. Which class is the *subclass*?

21. Which class is the *superclass*?

22. **tom** is an object of the **Student** class, not the **Person** class. The **Student** class does not have a **showAge** method.Why is it that this line: **tom.showAge();** does not cause a compile error?

23. Can a superclass inherit anything from a subclass?

24. Does a *subclass* have access to the **private** data from the *superclass*?

25. Does a *subclass* have access to the **protected** data from the *superclass*?

26. When an object of a *subclass* is instantiated both the *subclass* constructor and the *superclass* constructors are called, but which one is called first?

27. Look at program ***Java0909.java***. What is impractical about this program?

28. Look at program ***Java0911.java***. The keyword **super** is used to redirect the *age* information from the **Student** constructor to the **Person** constructor. Where in the must you place the **super** keyword?

29. If there are more than 2 levels of inheritance, can **super** be used to redirect data from the lowest subclass constructor to the highest superclass constructor?

30. What is the difference between *Multi-Level Inheritance* and *Multiple Inheritance*?

31. Refer to the previous question. Which of these is not allowed in Java?

32. Look at program ***Java0913.java***. This program displays the student’s *age* as *12* instead of *17*.

Why did it not call the inherited *getData* method from the *Person* class?

33. Refer to the previous question. In a situation where both the *superclass* and the *subclass* have a method with the same name, how do you call the *superclass’* method? (Hint: Look at program ***Java0914.java***.)

34. Can a superclass identifier declare a subclass object?

35. Look at program ***Java0916.java***. Why does this program not compile?

36. What is the *umbrella class* for the **Bug**, **Rock** and **Flower** classes?