|  |  |  |
| --- | --- | --- |
| **APCS Exposure Java** | **Exercises 12.01-07** | **Date:** |
| **Name:** | | **Period:** |

1. What are the 2 subtopics of *class interaction*?

2. What does *class interaction* require?

3. How is information passed to the constructor of a superclass?

4. How is information passed to the constructor of an attribute?

5. In the inheritance chapter, you saw that the superclass constructor was called before the subclass constructor. With composition, which comes first: the *composed* class constructor, or the *attribute* class constructor? (Hint: Look at the output of program ***Java1201.java***.)

6. Can one class be composed of several attribute classes?

7. Explain how the *Jack O’lantern Case Study*demonstrates both *inheritance* and *composition*?

8. In the *Jack O’lantern Case Study*, what is wrong with ***Stage 3***?

9. In the *Jack O’lantern Case Study*, how does ***Stage 4*** fix the problem of ***Stage 3***?

10. Explain how the *Train Case Study*demonstrates both *inheritance* and *composition*?

11. What improvement is made between ***Stage 1*** and ***Stage 2*** of the *Train Case Study*?

12. The final stage of the *Train Case Study* shows 4 classes in one file.

This was done for convenience; however, what is the proper style for a program with multiple classes?

13. Can an object contain an array?

14. Refer to the previous question. Is this an example of *composition*?

15. Look at programs ***Java1212.java*** and ***Java1213.java***. Both programs essentially do the same thing.

What is the difference between the 2 programs?

16. List 4 subclasses of **Actor**.

17. **BoxBug** is a subclass of \_\_\_\_\_\_\_.

18. **CrabCritter** and **ChameleanCritter** are subclasses of

19. We know the **Actor** class is involved with *inheritance* because it is the superclass for several other classes. Does the **Actor** class use any *composition*? Explain your answer.

20. Inside the **BoundedGrid** class is an attribute called **occupantArray**. What does it store?

21. What is the **Object** class?

22. What is the difference between *re-defining* and *newly-defining* a method?

22. Which method does the **Rock** class re-define?

23. Which method does the **Flower** class re-define?

24. Which method does the **Bug** class re-define?

25. Which method does the **Bug** class newly-define?