|  |
| --- |
| \* Multiple Choice: 2 points each |
| \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
|  |
| /\* 1. Which method on the StringBuilder class adds a String to the end? |
| \* a. concat |
| \* b. append |
| \* c. put |
| \* d. add |
| \*/ |
| char answer1 = 'b. append '; |
|  |
| /\* 2. Which of the following correctly instantiate an instance of a StringBuilder? |
| \* a. StringBuilder = new StringBuilder(); |
| \* b. String sb = new StringBuilder(); |
| \* c. StringBuilder() sb = new StringBuilder; |
| \* d. StringBuilder sb = new StringBuilder("Hello"); |
| \*/ |
| char answer2 = 'b. String sb = new StringBuilder(); '; |
|  |
| /\* 3. How would you pronounce `List<String>`? |
| \* a. String List |
| \* b. List of String |
| \* c. A bunch of Strings |
| \* d. String Array |
| \*/ |
| char answer3 = 'b. List of String '; |
|  |
| /\* |
| \* 4. What can go in the `<>` of a List? |
| \* a. Any Class |
| \* b. Any primitive data type |
| \* c. int, String, and custom classes |
| \* d. Any variable |
| \*/ |
| char answer4 = ‘c. int, String, and custom classes’; |
|  |
| /\* |
| \* 5. Which of the following is a correct instantiation of a List of Employee? |
| \* a. List<Employee> list = new List<Employee>(); |
| \* b. ArrayList<Employee> list = new ArrayList<Employee>(); |
| \* c. Collection<Employee> list = new List<Employee>(); |
| \* d. List<Employee> list = new ArrayList<Employee>(); |
| \*/ |
| char answer5 = ' d. List<Employee> list = new ArrayList<Employee>();'; |
|  |
| /\* |
| \* 6. Which of the following is best suited for storing customer names and unique customer IDs together in a collection? |
| \* a. List |
| \* b. Collection |
| \* c. Set |
| \* d. Map |
| \*/ |
| char answer6 = 'c. Set'; |
|  |
| /\* |
| \* 7. Which method on the Set class adds an entry to the Set? |
| \* a. put |
| \* b. push |
| \* c. add |
| \* d. append |
| \*/ |
| char answer7 = ' c. add'; |
|  |
| /\* |
| \* 8. Which of the following is not a pillar of OOP? |
| \* a. Inheritance |
| \* b. Classes |
| \* c. Polymorphism |
| \* d. Abstraction |
| \*/ |
| char answer8 = 'b. Classes '; |
|  |
| /\* |
| \* 9. What does OOP stand for? |
| \* a. Object Oriented Programming |
| \* b. Only Object Programming |
| \* c. One Ounce Preference |
| \* d. Object Oriented Preference |
| \*/ |
| char answer9 = 'a. Object Oriented Programming '; |
|  |
| /\* |
| \* 10. What is the value of inheriting from a class? |
| \* a. You can create code families. |
| \* b. You can avoid rewriting code by passing it down from a super class. |
| \* c. There isn't much value. |
| \* d. You can hide data so it's not used improperly. |
| \*/ |
| char answer10 = ' b. You can avoid rewriting code by passing it down from a super class. '; |
|  |
| /\* |
| \* 11. What is encapsulation? |
| \* a. Hiding the internal functionality of a class to ensure it is used properly. |
| \* b. Hiding data. |
| \* c. Extending a super class. |
| \* d. Moving common code to a super class to avoid code duplication. |
| \*/ |
| char answer11 = 'a. Hiding the internal functionality of a class to ensure it is used properly. '; |
|  |
| /\* |
| \* 12. What can access a class field marked as `private`? |
| \* a. Anything |
| \* b. Nothing |
| \* c. Only code within the class itself. |
| \* d. Only code within the same package. |
| \*/ |
| char answer12 = 'c. Only code within the class itself. '; |
|  |
| /\* |
| \* 13. Why do we mark class fields as `private` instead of `public`? |
| \* a. To make the code compile. |
| \* b. To make the data more accessible. |
| \* c. To hide our data. |
| \* d. To enforce encapsulation. |
| \*/ |
| char answer13 = 'd. To enforce encapsulation. '; |
|  |
| /\* |
| \* 14. Which of the following is an example of a checked exception? |
| \* a. FileNotFoundException |
| \* b. ArrayIndexOutOfBoundsException |
| \* c. ArithmeticException |
| \* d. NumberFormatException |
| \*/ |
| char answer14 = 'b. ArrayIndexOutOfBoundsException '; |
|  |
| /\* |
| \* 15. What keyword is used to allow a Class to implement an interface? |
| \* a. using |
| \* b. use |
| \* c. extends |
| \* d. implements |
| \*/ |
| char answer15 = 'd. implements '; |
|  |
| /\* |
| \* 16. When would you use an interface instead of an abstract class? |
| \* a. When you know `how` and `what` you want to happen. |
| \* b. When you know `what` you want to happen, but want to leave the `how` up to the implementing class. |
| \* c. When the code becomes too complex to put in an abstract class. |
| \* d. When you are building a dynamic application. |
| \*/ |
| char answer16 = 'b. When you know `what` you want to happen, but want to leave the `how` up to the implementing class.'; |
|  |
| /\* |
| \* 17. What does debugging mean in Eclipse? |
| \* a. It is another name for refactoring. |
| \* b. Removing bugs from code. |
| \* c. Stepping through code one line at a time to see what is really happening. |
| \* d. nothing |
| \*/ |
| char answer17 = 'c. Stepping through code one line at a time to see what is really happening.'; |
|  |
| /\* |
| \* 18. When debugging, what does `step over` do? |
| \* a. Ends the debugging process. |
| \* b. Skips to the next class. |
| \* c. Steps into the lines of the method on the current line. |
| \* d. Runs the method on the current line, but does not into the lines of the method. |
| \*/ |
| char answer18 = 'b. Skips to the next class. '; |
|  |
| /\* |
| \* 19. What is a Unit Test? |
| \* a. An application that tests network connectivity. |
| \* b. Code that tests our methods. |
| \* c. Code that tests for syntax mistakes. |
| \* d. An application that checks code quality. |
| \*/ |
| char answer19 = 'b. Code that tests our methods. '; |
|  |
| /\* |
| \* 20. Which annotation is used to denote a unit test? |
| \* a. @Testing |
| \* b. @Test |
| \* c. @UnitTest |
| \* d. @JUnit |
| \*/ |
| char answer20 = 'b. @Test '; |
|  |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| \* true or false: 2 points each |
| \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
|  |
| // 21. You should use a StringBuilder instead of a String when your String will change multiple times. |
| boolean answer21 = true ; |
|  |
| // 22. A generic can take any Object data type. |
| boolean answer22 = true; |
|  |
| // 23. An entry to a Map has both a key and a value. |
| boolean answer23 = true; |
|  |
| // 24. `List<String> list = new List<String>();` is a correct instantiation of a List of String. |
| boolean answer24 = false ; |
|  |
| // 25. `Map<Double, String>` the key is of type `Double`. |
| boolean answer25 = true; |
|  |
| // 26. The entries in a Set are ordered. |
| boolean answer26 = false; |
|  |
| // 27. A List is cannot contain duplicate entries. |
| boolean answer27 = false; |
|  |
| // 28. `public` members are accessible anywhere. |
| boolean answer28 = true; |
|  |
| // 29. `priave` members are accessible anywhere. |
| boolean answer29 = false; |
|  |
| // 30. Polymorphism is the ability for something to take many forms, or act differently in different forms. |
| boolean answer30 = true; |
|  |
| // 31. Class names should be written in the CamelCase convention. |
| boolean answer31 = true ; |
|  |
| // 32. A super class inherits from a sub class. |
| boolean answer32 = false; |
|  |
| // 33. OOP stands for Object Oriented Preference |
| boolean answer33 = false; |
|  |
| // 34. Encapsulation is data hiding. |
| boolean answer34 = true; |
|  |
| // 35. Interfaces and Abstract Classes are the same thing. |
| boolean answer35 = false; |
|  |
| // 36. Creating an interface to inherit will allow for more flexible and maintainable code. |
| boolean answer36 = false; |
|  |
| // 37. @Testing declares a method as a unit test. |
| boolean answer37 = false; |
|  |
| // 38. `assertEquals(var, 10);` will cause the test to pass if var is 9. |
| boolean answer38 = false; |
|  |
| // 39. Debugging allows you to step through code one line at a time. |
| boolean answer39 = true; |
|  |
| // 40. Unit Tests are an important industry standard. |
| boolean answer40 = true; |
|  |
|  |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| \* Essay Questions: 5 points each |
| \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
|  |
| /\* 41. What does TDD stand for and what does it entail? |
| \* Test Driven Development implements software programming that involves Unit Testing which refactors the source code. |
| \*/ |
|  |
| /\* 42. When would you use an interface? |
| \* Interface is used to implement abstraction and to implement multiple interfaces. |
| \*/ |
|  |
| /\* 43. What are the four pillars of OOP? |
| \* Abstraction, Encapsulation, Inheritance, and Polymorphism. |
| \*/ |
|  |
| /\* 44. Write a method named createStringCount that takes an int and returns a String that counts from zero to the |
| \* value of the int passed in. For example, if you call `createStringCount(5);` you should get back: "012345". |
| \* Use a StringBuilder in the method. |
| **public** **class** **question44new** **extends** **question44** {    **public** **String** **createStringCount**(**int** i) {  **int** **number** = 5;  **String** **createStringCount** = **Integer**.*toString*(number);        **return** createStringCount.toString(); }      **public** **static** **void** **main**(**String**[] args) {    **StringBuilder** **count** = **new** StringBuilder("012");  count.append( "345");    **System**.***out***.println(count);  }  } |
| \*/ |
|  |
|  |
| } |