|  |  |  |  |
| --- | --- | --- | --- |
| **Final Exam**  Completed: 15 | | | |
|  |  |  |  |
|  | | | |
| 1. | Suppose that your class does not use any package statement currently.   When is it necessary for you to use a package statement in your class? | | |
|  |  | (a) | If you want to refer to your class in classes belonging to packages other than the default package. |
|  |  | (b) | If you want to refer to your class in classes belonging to any package, including the default package. |
|  |  | (c) | If you want to refer to your class from any other class. |
|  |  | (d) | If you do not have a public keyword in front of your class definition. |
|  | | | |
| 2. | When is it necessary for you use the public keyword in front of your class definition? | | |
|  |  | (a) | If you want classes outside of that class' package to refer to it. |
|  |  | (b) | If you want objects instantiated from that class to refer to it. |
|  |  | (c) | If you want any other class to refer to it. |
|  |  | (d) | If you want objects instantiated from any other class to refer to it. |
|  | | | |
| 3. | Which of the following are valid import statements if you want to refer to a Java pre-defined class called 'ActionListener' defined under the java.applet.event package?   |  |  |  | | --- | --- | --- | |  | (i) | import java.applet.event.ActionListener; | |  | (ii) | import java.applet.event.\*; | |  | (iii) | import java.applet.\*; | |  | (iv) | import java.\*; | | | |
|  |  | (a) | All of the above |
|  |  | (b) | (i) only |
|  |  | (c) | (i) and (ii) only |
|  |  | (d) | (i), (ii), and (iii) only |
|  | | | |
| 4. | Suppose that you have a class called 'Test' whose definition is as shown below:   |  | | --- | | public class Test { String name = "undefined"; } |   Which of the following would be valid names for the file that contains this class, for the code to compile (select all that apply)?   |  |  |  | | --- | --- | --- | |  | (i) | Test.java | |  | (ii) | test.java | |  | (iii) | TestClass.java | |  | (iv) | You can name the file whatever you want. | | | |
|  |  | (a) | All of the above |
|  |  | (b) | (i) only |
|  |  | (c) | (i) and (ii) only |
|  |  | (d) | (i), (ii) and (iii) only |
|  | | | |
| 5. | What would happen if you don't use an explicit package statement in your class? | | |
|  |  | (a) | The class would become part of a default package |
|  |  | (b) | The class would not be part of any package |
|  |  | (c) | The class would become part of the most recent package you used |
|  |  | (d) | The code would not compile |
|  | | | |
| 6. | Suppose that you need to use the Java pre-defined class called 'Math' defined under the java.lang package in your class.  In order that the code compiles, which of the following is necessary in your class before you start using the Math class in your code? | | |
|  |  | (a) | Use an import statement: import java.lang.Math; |
|  |  | (b) | Use an import statement: import java.lang.\*; |
|  |  | (c) | Use an import statement: import java.\*; |
|  |  | (d) | No need to do anything, simply start using the Math class without any import statement or other qualification, since all classes from java.lang are imported into your class by default. |
|  | | | |
| 7. | What would the output of the following code be? Double d = null; System.out.println(" d = " + ( d == null? "null object" : d)); | | |
|  |  | (a) | d = null object |
|  |  | (b) | d = null |
|  |  | (c) | The code will not compile |
|  |  | (d) | The code will compile but will give a run time error |
|  | | | |
| 8. | Suppose that you have a class called Parent and a sub-class called Child, as shown below:   |  | | --- | | class Parent { String name; // default constructor in Parent Parent() { System.out.println(" in default constructor of parent"); } // Matching constructor in Parent Parent(String name) { this.name = name; System.out.println(" in matching constructor of parent"); } } class Child extends Parent { // default constructor in Child public Child(){ System.out.println(" in default constructor of child");} // specific constructor in Child public Child(String name) { super(name); System.out.println(" in specific constructor of child"); } } class TestParentChild { public static void main(String[] args){ Child child = new Child("John");} } |   What output would you expect to see when you run TestParentChild? | | |
|  |  | (a) | in matching constructor of parent in specific constructor of child |
|  |  | (b) | in default constructor of parent in matching constructor of parent in specific constructor of child |
|  |  | (c) | in default constructor of parent in specific constructor of child |
|  |  | (d) | in specific constructor of child in matching constructor of parent |
|  | | | |
| 9. | Using the Parent and Child classes of question 8 above, if TestParentChild is changed as shown below, what output would you expect if you run the new TestParentChild?   |  | | --- | | class TestParentChild { public static void main(String[] args){ Parent parent = new Child("John"); } } | | | |
|  |  | (a) | in matching constructor of parent in specific constructor of child |
|  |  | (b) | in default constructor of parent in matching constructor of parent in specific constructor of child |
|  |  | (c) | in default constructor of parent in specific constructor of child |
|  |  | (d) | The code would not compile |
|  | | | |
| 10. | Suppose that you have two classes: Person class and its sub-class, Customer. The Customer class has an attribute called 'orderValue'. Suppose that you have an object 'prs' of type Person that is currently pointing to a Customer type object. Suppose that you want to refer to the 'orderValue' attribute of the object in your expressions. Which of the following is a valid way of casting 'prs' to refer to this attribute? | | |
|  |  | (a) | ((Customer)prs).orderValue |
|  |  | (b) | (Customer)prs.orderValue |
|  |  | (c) | (Customer)(prs.orderValue) |
|  |  | (d) | Customer(prs).orderValue |
|  | | | |
| 11. | Which of the following is a valid beginning of a class definition for a class called Test that inherits from another class called Parent? | | |
|  |  | (a) | class Test inherits Parent |
|  |  | (b) | class Test extends Parent |
|  |  | (c) | class Test subclass Parent |
|  |  | (d) | class Test child Parent |
|  | | | |
| 12. | Suppose that you have two classes: Person class and its sub-class, Customer. Suppose that you have an object 'prs' of type Person that is currently pointing to a Customer type object. Which of the following is a valid way of testing whether 'prs' is currently pointing to a Customer data type? | | |
|  |  | (a) | prs objectOf Customer |
|  |  | (b) | prs pointsTo Customer |
|  |  | (c) | prs instanceof Customer |
|  |  | (d) | prs isA Customer |
|  | | | |
| 13. | What would the output of the following code be? Double d = 3.0; System.out.println(" d = " + d.toString()); | | |
|  |  | (a) | d = 3.0 |
|  |  | (b) | d = 3 |
|  |  | (c) | The code will not compile |
|  |  | (d) | The code will compile but will give run time errors |
|  | | | |
| 14. | Which of the following is true (select all that apply)?   |  |  |  | | --- | --- | --- | |  | (i) | A class that implements an interface must have a full method definition for every method of the interface class. | |  | (ii) | You cannot instantiate an object of a class that implements an interface class. | |  | (iii) | A class that implements an interface does not inherit class members of the interface class. | |  | (iv) | You cannot instantiate an object of an interface class. | | | |
|  |  | (a) | All of the above |
|  |  | (b) | (i) and (iv) only |
|  |  | (c) | (i), (ii) and (iv) only |
|  |  | (d) | (i), (iii) and (iv) only |
|  | | | |
| 15. | Which of the following is true (select all that apply)?   |  |  |  | | --- | --- | --- | |  | (i) | A class that extends an abstract class must have a full method definition for every method of the abstract class. | |  | (ii) | You cannot instantiate an object of a class that extends an abstract class. | |  | (iii) | A class that extends an abstract class does not inherit class members of the abstract class. | |  | (iv) | You cannot instantiate an object of an abstract class. | | | |
|  |  | (a) | All of the above |
|  |  | (b) | (i) and (iv) only |
|  |  | (c) | (ii), (iii) and (iv) only |
|  |  | (d) | (iv) only |
|  | | | |
| 16. | Which of the following is true regarding the 'protected' keyword? | | |
|  |  | (a) | It gives access to that class and its sub-classes only in any package - not to any other outsider, including objects instantiated from either the parent or the sub-classes. |
|  |  | (b) | It gives access to that class and its sub-classes only in that class' package - not to any other outsider, including objects instantiated from either the parent or the sub-classes. |
|  |  | (c) | It gives access to that class and its sub-classes and objects instantiated from either the parent or the sub-classes in any package - not to any other outsider. |
|  |  | (d) | It gives access to that class and its sub-classes in that class' package and objects instantiated from either the parent or the sub-classes in that class' package - not to any other outsider. |
|  | | | |
| 17. | Which of the following is a valid beginning of a class definition for an interface class called Test? | | |
|  |  | (a) | interface class Test |
|  |  | (b) | interface Test |
|  |  | (c) | interface Test class |
|  |  | (d) | class interface Test |
|  | | | |
| 18. | Which of the following is a valid beginning of a class definition for an abstract class called Test? | | |
|  |  | (a) | abstract Test |
|  |  | (b) | abstract Test class |
|  |  | (c) | class abstract Test |
|  |  | (d) | abstract class Test |
|  | | | |
| 19. | When is it necessary for you to override a parent class' method in the child class? | | |
|  |  | (a) | When you want the method in the child class to execute different instructions than what the one in the parent class does. |
|  |  | (b) | When you want the method in the child class to execute the same instructions as the one in the parent class. |
|  |  | (c) | None of the above |
|  |  | (d) | It is never necessary to override a parent class' method in the child class. |
|  | | | |
| 20. | Suppose that your method can potentially throw an exception.  Which of the following is true? | | |
|  |  | (a) | For the code to compile, you must either catch the exception (or its parent classes) in the method, or your method should have a throws clause that throws the exception or its parent classes, and this rule applies irrespective of what the type of the exception object is. |
|  |  | (b) | For the code to compile, you must either catch the exception (or its parent classes) in the method, or your method should have a throws clause that throws that exception (or its parent classes), except when the exception object is of RuntimeException type or its sub-classes. |
|  |  | (c) | It is never necessary to catch the exception or have a throws clause in your method; if you don't do either of these, the code will always compile but the exception will show up as an ugly message at run time. |
|  |  | (d) | None of the above. |
|  | | | |
| 21. | Suppose that you define your custom exception class called MyException as follows and save it in a file called MyException.java: class MyException extends Exception { }  Which of the following is true? | | |
|  |  | (a) | MyException.java will not compile because you must have at least one constructor in MyException class for it to compile. |
|  |  | (b) | MyException.java will not compile because you must have a constructor that takes a String argument in MyException class for it to compile. |
|  |  | (c) | MyException.java will compile, but you cannot throw exception objects of that type from the calling class; if you do, the calling class will not compile. |
|  |  | (d) | MyException.java will compile, but you can throw exception objects of that type from the calling class using MyException's default constructor only; if you try sending a String to MyException's constructor, the calling class will not compile. |
|  | | | |
| 22. | Suppose you have two exception classes: E1 and E2, with E2 being a subclass of E1, and E1 inheriting from the Exception class. If you have a try-catch statement with a single catch block that catches only exception objects of type E1, and your code throws an exception of type E2, what will happen? | | |
|  |  | (a) | The E2 exception object will not be caught in your catch block for E1 type. |
|  |  | (b) | The E2 exception object will be caught in your catch block for E1 type. |
|  |  | (c) | The E2 exception object will be caught by Java's default exception catching mechanism for all uncaught exceptions; therefore, the error message will show up as an ugly error message. |
|  |  | (d) | The code will compile but will give a ClassCastException at run time. |
|  | | | |
| 23. | Which method of the Exception class would you use to get the message string of the exception object? | | |
|  |  | (a) | getMessageString() |
|  |  | (b) | getMessage() |
|  |  | (c) | getExceptionMessage() |
|  |  | (d) | getErrorMessage() |
|  | | | |
| 24. | What block would use if you want the program to definitely execute instructions whether the exception is thrown or not? | | |
|  |  | (a) | The try block |
|  |  | (b) | The catch block |
|  |  | (c) | The finally block |
|  |  | (d) | The default block |
|  | | | |
| 25. | Suppose that you define your custom exception class called MyException as follows and save it in a file called MyException.java:   |  | | --- | | class MyException extends Exception { MyException (String str) { super(str); } } |   Suppose that you want to throw an exception with a message of "Invalid data" for a calling class.  Which of the following are valid for this purpose?   |  |  |  | | --- | --- | --- | |  | (i) | throw new MyException("Invalid data"); | |  | (ii) | MyException e = new MyException(); e.setMessage("Invalid data"); throw e; | |  | (iii) | MyException e = new MyException("Invalid data"); throw e; | | | |
|  |  | (a) | All of the above |
|  |  | (b) | (i) and (iii) only |
|  |  | (c) | (ii) and (iii) only |
|  |  | (d) | (i) and (ii) only |
|  | | | |
| 26. | Assume that you define a File object called tmpFile as follows: File tmpFile = new File("C:/Java/TestOutput.txt"); Which of the following are valid ways of instantiating a BufferedWriter object (select all that apply)?   |  |  |  | | --- | --- | --- | |  | (i) | BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(new FileOutputStream(tmpFile))); | |  | (ii) | BufferedWriter bw = new BufferedWriter(new FileWriter(tmpFile)); | |  | (iii) | BufferedWriter bw = new BufferedWriter(new FileWriter(tmpFile.getPath())); | |  | (iv) | BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(tmpFile)); | |  | (v) | BufferedWriter bw = new BufferedWriter(new FileOutputStream(tmpFile)); | | | |
|  |  | (a) | All of the above |
|  |  | (b) | (i) and (iv) only |
|  |  | (c) | (i), (ii) and (iii) only |
|  |  | (d) | (ii), (iii) and (v) only |
|  | | | |
| 27. | What would happen if you called the createNewFile() method on a file object, and the file already exists in the specified folder? | | |
|  |  | (a) | It would throw an exception of type FileAlreadyExistsException at run time and show up as an ugly error message. |
|  |  | (b) | It would cause a system crash - you will have to restart your machine to fix it. |
|  |  | (c) | It would create another file in that folder by preprending the words "Copy of" in front of the original file name. |
|  |  | (d) | It would not throw any exception; it would not attempt to create a new file with that name in that folder, and would use the existing file for subsequent instructions. |
|  | | | |
| 28. | Suppose that you instantiate an object of type FileWriter class and call the 'write()' method on it for writing to the file.   At the end of all such instructions, you do not use an explicit 'close()' statement on the FileWriter object.   What would happen when you try to compile and run this code? | | |
|  |  | (a) | The code would compile and run fine but the FileWriter object would not write anything to the file. |
|  |  | (b) | The FileWriter object would write to the file as usual - only thing is the memory won't be freed up for instantiating other i/o objects. |
|  |  | (c) | The code would not compile |
|  |  | (d) | The code would compile but would give a run time error. |
|  | | | |
| 29. | Which of the following are abstract base classes for character streams? | | |
|  |  | (a) | OutputStream and InputStream |
|  |  | (b) | StreamWriter and StreamReader |
|  |  | (c) | OutputStreamWriter and InputStreamReader |
|  |  | (d) | Writer and Reader |
|  | | | |
| 30. | Which of the following are abstract base classes for byte streams? | | |
|  |  | (a) | Writer and Reader |
|  |  | (b) | OutputStream and InputStream |
|  |  | (c) | StreamWriter and StreamReader |
|  |  | (d) | OutputStreamWriter and InputStreamReader |
|  | | | |
| 31. | Assume that you define a File object called tmpFileIn as follows: File tmpFileIn = new File("C:/Java/TestInput.txt"); Which of the following are valid ways of instantiating a BufferedReader object (select all that apply)?   |  |  |  | | --- | --- | --- | |  | (i) | BufferedReader br = new BufferedReader(new InputStreamReader(new FileInputStream(tmpFileIn))); | |  | (ii) | BufferedReader br = new BufferedReader(new FileReader(tmpFileIn)); | |  | (iii) | BufferedReader bw = new BufferedReader (new FileReader(tmpFileIn.getPath())); | |  | (iv) | BufferedReader bw = new BufferedReader (new InputStreamReader(tmpFileIn)); | |  | (v) | BufferedReader bw = new BufferedReader (new FileInputStream(tmpFileIn)); | | | |
|  |  | (a) | All of the above |
|  |  | (b) | (i) and (iv) only |
|  |  | (c) | (i), (ii) and (iii) only |
|  |  | (d) | (ii), (iii) and (v) only |
|  | | | |
| 32. | Suppose you want to start a new thread for performing specific instruction(s) (for example, your specific instruction could be a single statement: System.out.println("Hello");). For this, you define a class called MyThread that extends from the Thread class, as follows: class MyThread extends Thread { } Suppose that you write a separate class called TestThread which starts a thread of MyThread type that performs the specific instructions. How should your code be organized so that the net effect is to start a thread MyThread type and also to perform the specific instruction(s) you want? | | |
|  |  | (a) | In the TestThread class, instantiate an object of the MyThread class, call the start() method on it, then put your specific instruction(s) after this. |
|  |  | (b) | Define an init() method in MyThread, put all your specific instructions in it, then, inside TestThread, instantiate an object of the MyThread class, and call the start() method on it. |
|  |  | (c) | Define an initializeThread() method in MyThread, put all your specific instructions in it, then, inside TestThread, instantiate an object of the MyThread class, and call the start() method on it. |
|  |  | (d) | Define a run() method in MyThread, put all your specific instructions in it, then, inside TestThread, instantiate an object of the MyThread class, and then call the start() method on it. |
|  | | | |
| 33. | Suppose that you define a class called MyThread that extends from the Thread class, as follows:   |  | | --- | | class MyThread extends Thread { public void run() { for (int i = 1; i <= 10; i++) System.out.println(" i = " + i ); } } |   and define a test driver as follows:   |  | | --- | | class TestThread { public static void main(String[] args) { MyThread thr1 = new MyThread(); thr1.start(); MyThread thr2 = new MyThread(); thr2.start(); } } |   What would you expect when you run TestThread? | | |
|  |  | (a) | The output from 'thr1' would show up first for values of 'i' from 1 through 10, and then the values from 'thr2' would display for values of 'i' from 1 through 10. |
|  |  | (b) | The output from 'thr2' would show up first for values of 'i' from 1 through 10, and then the values from 'thr1' would display for values of 'i' from 1 through 10. |
|  |  | (c) | The exact output depends on availability of resources at the time of execution - the output from thr1 and thr2 could appear mixed up: some values from thr1, some from thr2, again some from thr1, and so on. |
|  |  | (d) | The system would crash - you would have to restart your machine to fix it. |
|  | | | |
| 34. | What is the default type for a thread object that spawns from a daemon thread? | | |
|  |  | (a) | Daemon thread |
|  |  | (b) | User thread |
|  |  | (c) | Runnable thread |
|  |  | (d) | Default thread |
|  | | | |
| 35. | What is the default type for a thread object that doesn't spawn from a daemon thread? | | |
|  |  | (a) | Daemon thread |
|  |  | (b) | User thread |
|  |  | (c) | Runnable thread |
|  |  | (d) | Default thread |
|  | | | |
| 36. | What are the ways you can define your thread class (select all that apply)?   |  |  |  | | --- | --- | --- | |  | (i) | Extend from the Thread class | |  | (ii) | Implement the Runnable interface | |  | (iii) | Implement the Thread interface | |  | (iv) | Extend from the Runnable class | | | |
|  |  | (a) | All of the above |
|  |  | (b) | (i) and (ii) only |
|  |  | (c) | (i) and (iv) only |
|  |  | (d) | (ii) and (iii) only |
|  | | | |
| 37. | Suppose that you have a thread running and you want to pause its execution temporarily.   Which of the following methods would you call on the thread object for this purpose? | | |
|  |  | (a) | The sleep() method |
|  |  | (b) | The interrupt() method |
|  |  | (c) | The pause() method |
|  |  | (d) | The break() method |