Multiple Choice
Identify the choice that best completes the statement or answers the question.

1. Inheritance is the process by which a new class – known as a _______ - is created from another class, called the _______.
   a. base class, derived class  
   b. derived class, base class  
   c. inherited class, base class  
   d. base class, inherited class

2. Inheritance promotes code ___________.
   a. reinvention  
   b. reuse  
   c. repeats  
   d. all of the above

3. The keyword extends indicates:
   a. encapsulation  
   b. polymorphism  
   c. inheritance  
   d. none of the above

4. What does a derived class automatically inherit from the base class?
   a. instance variables  
   b. static variables  
   c. public methods  
   d. all of the above

5. If the final modifier is added to the definition of a method, this means:
   a. The method may be redefined in the derived class.  
   b. The method may be redefined in the sub class.  
   c. The method may not be redefined in the derived class.  
   d. None of the above.

6. The special syntax for invoking a constructor of the base class is:
   a. super()  
   b. base()  
   c. parent()  
   d. child()

7. An object of a derived class has the type of the derived class, and it also has the type of the base class, and more generally, has the type of every one of its ________ classes.
   a. descendant  
   b. child  
   c. ancestor  
   d. sub

8. In using the keyword this() in place of super(), the invocation of this() must be the ________ action taken by the constructor.
   a. first  
   b. last  
   c. it does not matter  
   d. none of the above

9. The class ________ is an ancestor class of all Java classes.
   a. String  
   b. Object  
   c. Math  
   d. JFrame

10. The Object class contains the method:
    a. getClass()  
    b. toString()  
    c. equals()  
    d. all of the above

11. The equals method for a class should have ________ as the type of its one parameter.
    a. String  
    b. Integer  
    c. "Object"
12. The principals of object oriented programming include:
   a. encapsulation
   b. inheritance
   c. polymorphism
   d. all of the above

13. ________ binding refers to the method definition being associated with the method invocation when the code is compiled.
   a. Dynamic
   b. Late
   c. Early
   d. None of the above

14. ________ refers to the ability to associate many meanings to one method name by means of the late binding mechanism.
   a. Inheritance
   b. Encapsulation
   c. Polymorphism
   d. None of the above

15. A method marked as final means the compiler uses ________ binding.
   a. dynamic
   b. early
   c. late
   d. none of the above

16. Java does not use late binding for methods marked as:
   a. final
   b. static
   c. private
   d. all of the above

17. Assigning an object of a derived class to a variable of a base class is called:
   a. static binding
   b. dynamic binding
   c. upcasting
   d. downcasting

18. Assigning an object of an ancestor class to a descendent class is called:
   a. static binding
   b. dynamic binding
   c. upcasting
   d. downcasting

19. You cannot create an object using a/an:
   a. superclass constructor
   b. subclass constructor
   c. ancestor class constructor
   d. abstract class constructor

20. An abstract method cannot be modified by:
   a. public
   b. protected
   c. private
   d. none of the above

21. A class that has at least one abstract method is called an:
   a. concrete class
   b. encapsulated class
   c. abstract class
   d. private class

22. A class with no abstract methods is called a
   a. concrete class
   b. encapsulated class
   c. abstract class
   d. private class

23. **try** blocks contain code that could possibly:
   a. handle an exception
   b. throw an exception
   c. catch an exception
   d. display an exception

24. The **Exception** class belongs to the package:
   a. java.io
   b. java.lang
25. The execution of a **throw** statement is referred to as:
   a. catching a block
   b. trying a block
   c. handling an exception
   d. throwing an exception

26. The **catch** block has ________ parameters.
   a. zero
   b. one
   c. two
   d. three

27. A __________ block should immediately follow a **try** block.
   a. try
   b. catch
   c. fail
   d. final

28. If a method does not catch an exception, then it must at least warn programmers that any invocation of the method might possibly throw an exception. This warning is called a/an:
   a. Exception handler
   b. **throws** clause
   c. **try** block
   d. **catch** block

29. Which circumstance is an exception to the **catch or declare** rule?
   a. Exceptions that result from errors of some sort.
   b. Exceptions that are descendents of the class **RuntimeException**.
   c. Both A and B
   d. None of the above

30. Exceptions that are subject to the **catch or declare** rule are called:
   a. Checked exceptions
   b. Unchecked exceptions
   c. Fatal exceptions
   d. All of the above

31. A runtime exception is a/an:
   a. checked exception
   b. unchecked exception
   c. offending exception
   d. none of the above

32. All exceptions are descendents of the class:
   a. **Throwable**
   b. **Catchable**
   c. **Tryable**
   d. **Blockable**

33. Exception handling is an example of a programming methodology known as:
   a. structured programming
   b. object oriented programming
   c. goto programming
   d. event-driven programming

34. A __________ block executes regardless of whether an exception occurs.
   a. **final**
   b. **finally**
   c. **catch**
   d. none of the above

35. **ArrayIndexOutOfBoundsException** is a descendent of the class **RuntimeException**. This means:
   a. the exception must be caught
   b. a **finally** block must be included
   c. the exception does not have to be explicitly caught
   d. none of the above

36. An exception is caught in a ________ block.
True/False
Indicate whether the statement is true or false.

___ 37. A derived class contains only public instance variables and public methods from the base class.

___ 38. A derived class is a class defined by adding instance variables and methods to an existing class.

___ 39. Overriding is when a derived class redefines a method from the base class.

___ 40. You may substitute the keyword this() for super() to call a constructor of the derived class.

___ 41. An instance variable (or method) that is private in a base class is accessible by name in the definition of a method in any other class.

___ 42. Polymorphism refers to the ability to associate many meanings to one method through dynamic binding.

___ 43. Java allows an instance of an abstract class to be instantiated.

___ 44. Late binding refers to the method definition being associated with the method invocation when the method is invoked at run time.

___ 45. The final modifier is included before the definition of the method, then the method can be redefined in a derived class.

___ 46. The type of the variable naming an object determines which method names can be used in an invocation with that calling object.

___ 47. An abstract class is a class that has some methods without complete definitions.

___ 48. An abstract method serves as a placeholder for a method that must be defined in all derived classes.

___ 49. The throw operator causes a change in the flow of control.

___ 50. When an exception is thrown, the code in the surrounding try block continues executing and then the catch block begins execution.

___ 51. The two most important things about an exception object are its type and the message that it carries in an instance variable of type String.

___ 52. A program can catch multiple exceptions.

___ 53. The compiler does not complain when the catch or declare rule is ignored.

___ 54. You can not place a try block and its following catch blocks inside a larger try block or inside a larger catch block.

___ 55. The finally block contains code to be executed whether or not an exception is thrown in a try block.

Problem
56. [20 pts] Java graphics considers all drawable objects as living inside a bounding rectangle. The rectangle has a position defined by the upper-left corner \((x, y)\). The rectangle has a size defined by the width and height \((w, h)\) of the rectangle. All values are measured in integer pixel counts. Consider the abstract class \texttt{Shape} that contains the data fields: \(x, y, w,\) and \(h\). It also contains the abstract method: \texttt{double area()} . There are two descendent classes from \texttt{Shape}, namely \texttt{Rect} and \texttt{Oval}.

Write minimal classes for \texttt{Shape}, \texttt{Rect}, and \texttt{Oval}. Include the above mentioned data fields, constructors (complete only), and the method \texttt{area()}. No need for any other methods. The formula for the area of a rectangle is the product of its width and height. The formula for the area of an oval (or ellipse) is \(\pi \) (pi) times the product of the radii along the major and minor axes, from the center of the oval to the edge of the oval (which happens to interesect with the outline of the bounding rectangle).
## CS26 Exam 1
### Answer Section

### MULTIPLE CHOICE

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>ANS: B</td>
<td>PTS: 1</td>
</tr>
<tr>
<td>2.</td>
<td>ANS: B</td>
<td>PTS: 1</td>
</tr>
<tr>
<td>3.</td>
<td>ANS: C</td>
<td>PTS: 1</td>
</tr>
<tr>
<td>4.</td>
<td>ANS: D</td>
<td>PTS: 1</td>
</tr>
<tr>
<td>5.</td>
<td>ANS: C</td>
<td>PTS: 1</td>
</tr>
<tr>
<td>6.</td>
<td>ANS: A</td>
<td>PTS: 1</td>
</tr>
<tr>
<td>7.</td>
<td>ANS: C</td>
<td>PTS: 1</td>
</tr>
<tr>
<td>8.</td>
<td>ANS: A</td>
<td>PTS: 1</td>
</tr>
<tr>
<td>9.</td>
<td>ANS: B</td>
<td>PTS: 1</td>
</tr>
<tr>
<td>10.</td>
<td>ANS: D</td>
<td>PTS: 1</td>
</tr>
<tr>
<td>11.</td>
<td>ANS: B</td>
<td>PTS: 1</td>
</tr>
<tr>
<td>12.</td>
<td>ANS: D</td>
<td>PTS: 1</td>
</tr>
<tr>
<td>13.</td>
<td>ANS: C</td>
<td>PTS: 1</td>
</tr>
<tr>
<td>14.</td>
<td>ANS: C</td>
<td>PTS: 1</td>
</tr>
<tr>
<td>15.</td>
<td>ANS: B</td>
<td>PTS: 1</td>
</tr>
<tr>
<td>16.</td>
<td>ANS: D</td>
<td>PTS: 1</td>
</tr>
<tr>
<td>17.</td>
<td>ANS: C</td>
<td>PTS: 1</td>
</tr>
<tr>
<td>18.</td>
<td>ANS: D</td>
<td>PTS: 1</td>
</tr>
<tr>
<td>19.</td>
<td>ANS: D</td>
<td>PTS: 1</td>
</tr>
<tr>
<td>20.</td>
<td>ANS: C</td>
<td>PTS: 1</td>
</tr>
<tr>
<td>21.</td>
<td>ANS: C</td>
<td>PTS: 1</td>
</tr>
<tr>
<td>22.</td>
<td>ANS: A</td>
<td>PTS: 1</td>
</tr>
<tr>
<td>23.</td>
<td>ANS: B</td>
<td>PTS: 1</td>
</tr>
<tr>
<td>24.</td>
<td>ANS: C</td>
<td>PTS: 1</td>
</tr>
<tr>
<td>25.</td>
<td>ANS: D</td>
<td>PTS: 1</td>
</tr>
<tr>
<td>26.</td>
<td>ANS: B</td>
<td>PTS: 1</td>
</tr>
<tr>
<td>27.</td>
<td>ANS: B</td>
<td>PTS: 1</td>
</tr>
<tr>
<td>28.</td>
<td>ANS: B</td>
<td>PTS: 1</td>
</tr>
<tr>
<td>29.</td>
<td>ANS: C</td>
<td>PTS: 1</td>
</tr>
<tr>
<td>30.</td>
<td>ANS: A</td>
<td>PTS: 1</td>
</tr>
<tr>
<td>31.</td>
<td>ANS: B</td>
<td>PTS: 1</td>
</tr>
<tr>
<td>32.</td>
<td>ANS: A</td>
<td>PTS: 1</td>
</tr>
<tr>
<td>33.</td>
<td>ANS: D</td>
<td>PTS: 1</td>
</tr>
<tr>
<td>34.</td>
<td>ANS: B</td>
<td>PTS: 1</td>
</tr>
<tr>
<td>35.</td>
<td>ANS: C</td>
<td>PTS: 1</td>
</tr>
<tr>
<td>36.</td>
<td>ANS: B</td>
<td>PTS: 1</td>
</tr>
</tbody>
</table>

### TRUE/FALSE

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>37.</td>
<td>ANS: F</td>
<td>PTS: 1</td>
</tr>
<tr>
<td>38.</td>
<td>ANS: T</td>
<td>PTS: 1</td>
</tr>
</tbody>
</table>
56. **ANS:**
```java
public abstract class Shape  // 2 pts
{
    protected int x, y;       // 2 pts
    protected int w, h;

    protected Shape(int x, int y, int w, int h)
    {
        this.x = x;            // 3 pts
        this.y = y;
        this.w = w;
        this.h = h;
    }

    public abstract double area();  // 3 pts
}

public class Rect extends Shape  // 2 pts
{
    public Rect(int x, int y, int w, int h)
    {
        super(x, y, w, h);     // 2 pts
    }

    public double area()        // 2 pts
    {
        return this.w * this.h;
    }
}
```
public class Oval extends Shape // 1 pts
{
    public Oval(int x, int y, int w, int h)
    {
        super(x, y, w, h);
    }

    public double area() // 3 pts
    {
        return (Math.PI / 4.0) * (this.w * this.h);
    }
}

PTS: 20