5. Present types of inheritance in object-oriented programming. Which inheritance is not allowed in Java?
6. Present the difference between inheritance and composition in object-oriented programming.
7. Present the concept of polymorphism in object-oriented programming. Why do we care about polymorphism?
8. Present types of polymorphism with examples in Java.
9. Create an array of polymorphic objects in Java to show that objects of child classes can be treated as if they are objects of the parent class.
10. Present the concept of abstraction in object-oriented programming. How can we obtain abstraction in Java?
11. What are abstract classes? Present rules of abstract classes in Java.
12. What are abstract methods? Present rules of abstract methods in Java.
13. What are interfaces? How to implement an interface in Java?
14. Create an example to demonstrate the use of interface for multiple inheritance in Java.
15. Given the source code as follows:

```
public abstract class Cat {
  public String cName;
  public void chaseMouse() {
    System.out.print("Chasinggg mouse");
  }
  public abstract void meow() {
    System.out.print("Meowwwww");
  }
}
```

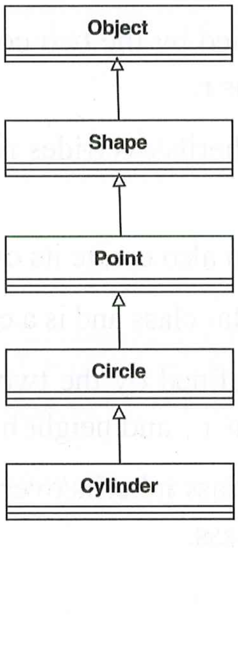
- Is there any problem with the source code above?
  - If yes, describe how to fix it.
  - If no, explain the reason.

16. Given the source code as follows:

```
public class MyCat {
  public static void main(String [] args) {
    Cat mCat = new Cat(); // (1)
    mCat.cName = "Peter"; // (2)
    mCat.meow(); // (3)
  }
}
```

- Explain the meaning of the code line (1), (2) and (3).
- Provide a better way to write the source codes above.

17. Design and implement the following multi-level inheritance tree in Java:



- `Object` class is the root class in Java.
- `Shape` is an abstract class and is a child of the `Object` class. The `Shape` class contains two regular methods and one abstract method:
  - A regular method `calArea()` returns the area of the shape.
  - A regular method `calVolume()` returns the volume of the shape.
  - An abstract method `getName()` returns the name of the shape.
- `Point` is a regular class and is a child of the `Shape` class:
  - A point is defined by the two coordinates `(x, y)`.
  - `Point` class inherits/overrides regular methods of `Shape` class and implements an abstract method of `Shape` class.
  - `Point` class can also create its own methods.
- `Circle` is a regular class and is a child of `Point` class:
  - A circle is defined by the two coordinates `(x, y)` of the center and radius r.
  - `Circle` class inherits/overrides regular methods of Point class.
  - `Circle` class can also create its own methods.
- `Cylinder` is a regular class and is a child of the `Circle` class:
  - A cylinder is defined by the two coordinates `(x, y)` of the center, radius `r`, and height `h`.
  - `The Cylinder` class inherits/overrides regular methods of the `Circle` class.

  - o `The Cylinder` class can also create its own methods.

18. Create a `ShapeTest` Java program to check the inheritance relationship of `Point`, `Circle`, and `Cylinder` with the `Shape` class in the inheritance tree in question 16 as follows:
    - Use the polymorphism concept to create an array of objects of the three classes: `Point`, `Circle`, and `Cylinder`.
    - Browse the created polymorphic array to perform the four following operations for each element of the array:
      - o Get the name of the object to see if it is a `Point`, a `Circle`, or a `Cylinder`.
      - o Calculate the area of the object.
      - o Calculate the volume of the object.
      - o Display the name, area, and volume of each object to the standard output.

19. Redo questions 16 and 17, but `Shape` is an interface and contains three abstract methods: `calArea()`, `calVolume()`, and `getName()`.