public class super1 {
    public void do1(super1 param) {}
}
public class sub extends super1 {
    public void do1(super1 param) {}
}

super1 a;
a = new sub();
public class super1 {
    public void do1(super1 param) {} 
}

public class sub extends super1 {
    public void do1(super1 param) {} 
}

super1 a;
a = new sub();

a.do1(a);

Which method is called?
public class super1 {
    public void do1(super1 param) {} 
}
public class sub extends super1 {
    public void do1(super1 param) {} 
}

super1 a;

a = new super1(a);
a.do1(a);

Which method is called?
public class super1 {
    public void do1(super1 param) {} 
}

public class sub extends super1 {
    public void do1(super1 param) {} 
}

super1 a;
a = new sub();
a = (super1) a;
a.do1(a);

Which method is called?
public class super1 {
    public void do1(super1 param) {}
}

public class sub extends super1 {
    public void do1(super1 param) {}
    public void do1(sub param) {}
}

super1 a;
super1 b;
a = new super1();
b = new sub();
a.do1(b);

Which method is called?
public class super1 {
    public void do1(super1 param) {} 
}

public class sub extends super1 {
    public void do1(super1 param) {} 
    public void do1(sub param) {} 
}

super1 a; 
sub b; 

a = new sub(); 
b = new sub();
a.do1(b);

Which method is called?
Which method is called?
public class super1 {
}

public class sub extends super1 {
    public super1() {
    }
    public sub(int i) {
    }
}

super1 a;

a = new sub();

Which constructor(s) is/are called?
public class super {
    public abstract int method(int j) {return 0;}
}

What’s the problem(s) with this?
public abstract class super1 {
    public abstract int method(int j);}

public class sub extends super1 {

    public int method() {return 1;}

}

What’s the problem with this?
public abstract class super1 {
    public abstract int method(int j);
}

gpublic abstract class sub extends super1 {
    public int method(int j) {return 1;}
}

gpublic class subsub extends sub {
}

Is this okay?
public interface inter1 {

    public int getInt(String s) {
        return 1;
    }
}

What’s wrong with this?
public interface inter1 {

    public int getInt(String s);

}

public class one extends two implements inter1 {

}

This compiles. Why?
• Texts can be CDs, books, newspapers or single sheets, and every text is one of these.

• What does the above suggest to you?
• How would you improve this code:
  
  ```java
  if (code == CDCode) {doSomething}
  else if (code == BookCode) {something else}
  else if (code == NewspaperCode) {}
  else if (code == singleSheetCode) {}
  ```
All texts have a title and a replacement cost
Books and CDs are also rectangulars - a rectangular is something that can be queried for its height, width and thickness. (There can be rectangulars which are not texts.)

• (Texts are rectangulars?)
Every text has a set of standard methods that are used to access and display it. Where are these defined?
All your text instances will be stored in a library. Explain a way to make sure that this actually happens - that **ALL** text instances are stored in the library.
• An airplane is a tail, an engine and a fuselage and two wings.

• Therefore:

• the class airplane should extend/ implement the interfaces tail, engine, fuselage and wing.

• **Explain what is wrong with this reasoning.**
• An airplane is a tail, an engine and a fuselage and two wings.

• Therefore:

• the class airplane should extend /implement the interfaces tail, engine, fuselage and wing.

• How should this probably be implemented?
public class BigException extends Exception {
}

public class one {
    public void MyMethod() {
        throw new BigException()
    }
}

This does not compile. Why not?
public class BigException extends Exception {

}

public class one {

    public void MyMethod() {
        throw new BigException()
    }

}

This does not compile. Tell me one way to make it compile
public class BigException extends Exception {
}

does not compile. Tell me a second way to make it compile
public class BigException extends Exception {
}

public class one {

    public void MyMethod() {
        throw new BigException()
    }
}

This does not compile. Tell me a third way to make it compile
public class super1 {
    public super1(int i) {} }

public class sub extends super1 {
    public sub() {};
    public sub (int i) {};
}

super1 a;

a = new sub(3);

Why is there a problem?
public class super1 {
    public super1(int i) {}
}
public class sub extends super {[…}
    public sub() {}
    public sub (int i) {};
}

super1 a;
a = new sub(3);

How do we arrange to call both (int) constructors?
public abstract class super1 {
    public abstract int method(int j);}

public class sub extends super1 {

    public int method(int j) {return 1;}}

super1 A = new super1();

What’s the problem with this?