11.4 How do you explicitly invoke a superclass's constructor from a subclass? Provide an example.

11.4 Answer

To invoke the superclass's constructor from a subclass, using super method, you have to write this method as "super();" at the first line in the subclass's constructor because the superclass should be initialized before subclass.

11.5 How do you invoke an overridden superclass method from a subclass? Provide an example.

11.5 Answer

To invoke an overridden superclass method from a subclass, you instantiate the subclass by using the new operator, but it returns a superclass's reference variable. For example, if there are a superclass named Super and a subclass named Sub, you can write "Super object = new Sub();" to invoke an overridden method from the subclass.

11.6 Explain the difference between method overloading and method overriding.

11.6 Answer

Method overloading is to define two or more methods with a same name, a different return value and different number/type of arguments within the same class.

On the other hand, method overriding is to define methods with the same signature both in the superclass and a subclass. A method defined in a subclass overrides the method in the superclass.

11.7 If a method in a subclass has the same signature as a method in its superclass with the same return type is the method overridden or overloaded?

11.7 Answer

It is the method overridden.

11.8 If a method in a subclass has the same signature as a method in its superclass with a different return type, will this be a problem?

11.8 Answer

This will be a problem, even though the program will work. This is because method overriding won't come into effect in this case, so that overriding makes no sense.

11.9 If a method in a subclass has the same name as a method in its superclass with different parameter types is the method overridden or overloaded?

11.9 Answer

It is the method overloaded.

11.17 There are three errors in the code below. Identify them.

```
ArrayList list = new ArrayList();
list.add ("Denver");
list.add("Austin");
list.add(new java.util.Date());
String city = list.get(0);
list.set(3, "Dallas");
System.out.println(list.get(3));
```

11.17 Answer

First error is "String city = list.get(0);" that is a cast error. "list.get(0)" is a reference type, so that you need to convert that to String such "String city = (String) list.get(0);".

Second error is "list.set(3, "Dallas");". This is trying to set "Dallas" to index 3 of the list, although only indices 0 to 2 have elements. You can change that to "list.add(3, "Dallas");" or "list.set(2, "Dallas");".

Third error is "System.out.println(list.get(3));". Just like the second error, this is trying to get the element of index 3, although index 3 is empty. You can change that to "System.out.println(list.get(2));".

11.22 How do you prevent a class from being extended? How do you prevent a method from being overridden?

11.22 Answer

If you want to prevent a class from being extended, you need to use the access modifier "private" for its fields or methods. Or, if you want to prevent a method from being overridden, you need to use "private" or "final" for the method.