1

polymorphism is referencing super class object toward sub class object

True

Polymorphism in OOP occurs when a super type references a sub type object

2

False

If the method overrides one of its superclass's methods, you can invoke the overridden method through the use of the keyword super.

If the super class constructor has no parameters, you do not need to use the super keyword to call the super class constructor in the subclass constructor, and the translater will automatically call the super class's parameterless constructor.

3

True

Abstract classes do not necessarily contain abstract methods, but classes that contain abstract methods must be declared as abstract classes.

4

False

The void keyword specifies that a method have not a return value, no mandatory have any parameter.

5

True

The dot ( . ) is used to access the object's attributes and methods. To call a method in Java, write the method name followed by a set of parentheses (), followed by a semicolon

6

override corresponding method number of parameters

False

Method overloading in JAVA with different number of parameters, different order or different types has these rules:

1. The method name is the same, the parameter type is the same, only the return value is different, this does not constitute overloading

2. Only the names of the formal parameters are different, which does not constitute method overloading.

3. Like ordinary methods, constructors can also be overloaded.

7

automatically make a reference variable of a subclass type point to an object of its superclass

False

We can use Referencing variable using superclass reference or Referencing variable using subclass reference

8

derived

class Secret {

public void tell() {

System.out.println("This is Secret");

}

}

class Mystery extends Secret {

public void tell() {

System.out.println("This is Mystery");

}

}

9

instance variable

The reference this is not only used to refer to the methods, also variable.

If the instance variable name of the method or constructor is the same as the local variable name, you can use this keyword to access the instance variable.

class ThisInstanceVariable{

String str;

ThisInstanceVariable(String str){

this.str = str;

}

public void print(){

System.out.println(str);

}

public static void main(String[] args){

ThisInstanceVariable instStr= new ThisInstanceVariable("This is a Instance String");

instStr.print();

}

}

10

D

getText

11

B

char

12

A

control is returned to method A

13

class members consist of all of the following except

C

pre-defined methods

14

D

all of the above are part of a method call

15

actual parameter

The value passed into a method by a method call