COMP2396B Tutorial 5

Inheritance, overloading, abstract classes

1. When a method is called on an object reference, the \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ of the method for that object type will be called —In other words, the \_\_\_\_\_\_ one in the inheritance tree wins!
2. It is possible to call an overridden method of the superclass using the keyword \_\_\_\_\_.
3. A superclass can choose whether or not it wants a subclass to inherit a particular member by the

access level assigned to that particular member .

—\_\_\_\_\_ members are inherited.

—\_\_\_\_\_\_ members are not inherited.

1. Method overloading is nothing more than having 2 or more methods with the same \_\_\_\_ but different \_\_\_\_\_\_\_\_\_\_\_ —It has nothing to do with inheritance and polymorphism.

—For overloaded methods

—Argument lists \_\_\_\_ be different.

—Return types \_\_\_ be different.

—Access levels \_\_\_ be different.

1. An abstract class can be used as a \_\_\_\_\_\_\_\_\_ (e.g., using it as a polymorphic argument or return type, or to make a polymorphic array).
2. Besides classes, \_\_\_\_\_\_ can also be marked as abstract.
3. An abstract class means the class must be \_\_\_\_\_\_\_, whereas an abstract method means the method must be \_\_\_\_\_\_\_\_.
4. An abstract method has no \_\_\_\_\_\_\_ , just ends with a semicolon.
5. A class must be marked as abstract if it has \_\_\_\_\_\_\_\_ abstract method. It is illegal to have an abstract method in a non-abstract class!

— An abstract class, on the other hand, can have \_\_\_\_\_\_\_\_ abstract and non-abstract methods.

1. A concrete class in the inheritance tree must \_\_\_\_\_\_\_\_\_ the abstract methods from its superclass.

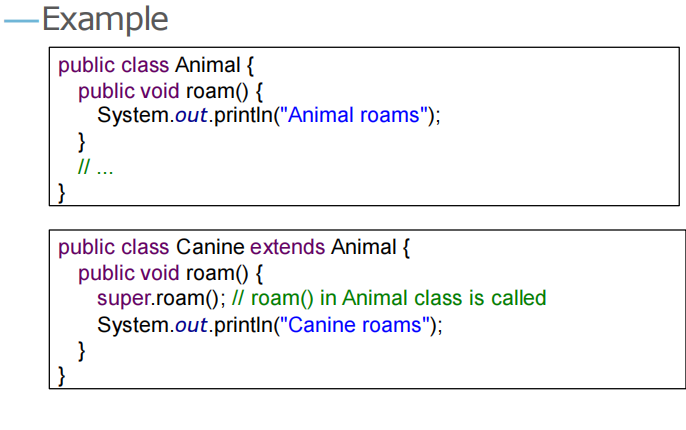
**Inheritance**

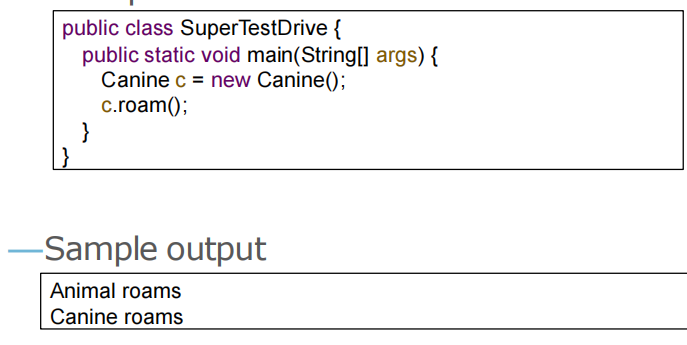
What is the output of the following program? It has no compile-time or run-time error.

|  |
| --- |
| **class** A {  **int** n;  **public** A() {  **this**.n = 1;  }  **public** **void** calculate() {  **this**.n = 4 \* **this**.n;  }  **public** **void** print() {  calculate();  System.***out***.println("In A: " + **this**.n);  }  }  **class** B **extends** A {  **int** n;  **public** B() {  **this**.n = 10;  }  **public** **void** calculate() {  **this**.n = 4 \* **super**.n;  }  **public** **void** print() {  **this**.calculate();  System.***out***.println("In B: " + **this**.n);  }  }  **class** C **extends** B {  **int** n;  **public** C() {  **this**.n = 100;  }  **public** **void** calculate() {  **this**.n = 4 \* **this**.n;  }  **public** **void** print() {  **super**.print();  System.***out***.println("In C: " + **this**.n);  }  }  **public** **class** Main {  **public** **static** **void** main(String[] args) {  A x1 = **new** A();  x1.print();  B x2 = **new** B();  x2.print();  C x3 = **new** C();  x3.print();  }  } |

Answer:

1. When a method is called on an object reference, the most specific version of the method for that object type will be called —In other words, the lowest one in the inheritance tree wins!
2. It is possible to call an overridden method of the superclass using the keyword super.



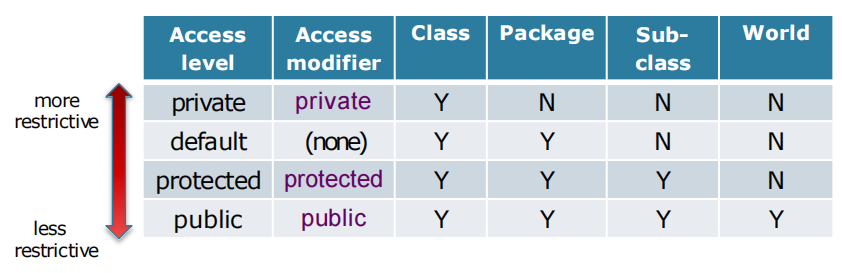


1. A superclass can choose whether or not it wants a subclass to inherit a particular member by the

access level assigned to that particular member .

—public members are inherited.

—private members are not inherited.



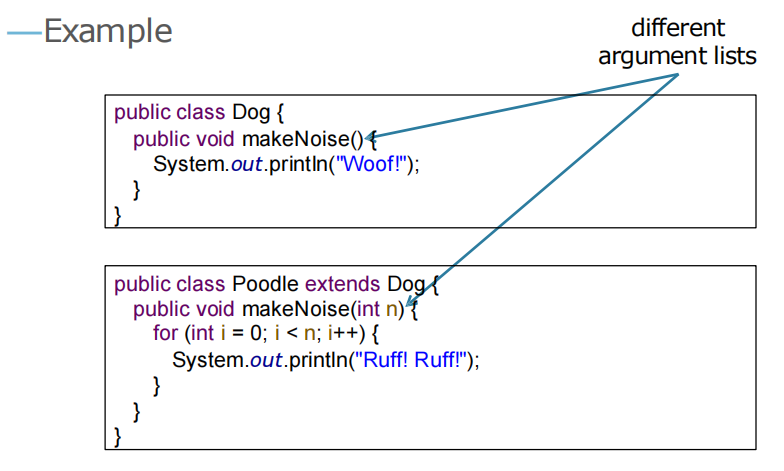
1. Method overloading is nothing more than having 2 or more methods with the same name but different argument lists —It has nothing to do with inheritance and polymorphism.

—For overloaded methods

—Argument lists must be different.

—Return types can be different.

—Access levels can be different.



1. An abstract class can be used as a reference type (e.g., using it as a polymorphic argument or return type, or to make a polymorphic array).
2. Besides classes, methods can also be marked as abstract.
3. An abstract class means the class must be extended, whereas an abstract method means the method must be overridden.
4. An abstract method has no method body, just ends with a semicolon.



1. A class must be marked as abstract if it has at least one abstract method. It is illegal to have an abstract method in a non-abstract class!

— An abstract class, on the other hand, can have either or both abstract and non-abstract methods.

1. A concrete class in the inheritance tree must implement all the abstract methods from its superclass.