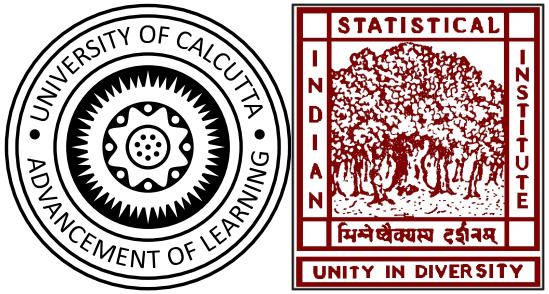
Introduction to JAVA Programming

Rathindra Nath Dutta

Junior Research Fellow
Advanced Computing & Microelectronics Unit
Indian Statistical Institute, Kolkata

June 15, 2018



Outline

Inheritance

- A way to incorporate and extend the definition of one class(*superclass*) into another class(*subclass*)
- Hierarchical definition of classes reduces code size and makes it more manageable
- The `extends` keyword is used to inherit a class in Java

```
class BaseClass {
    //class body
}
```

```
class DerivedClass extends BaseClass {
    //class body
}
```

- Java does not allow mutiple inheritance: multiple superclasses inherited into a single subclass
- Multilevel inheritance is allowed
- All inheritable members (both data & methods) of base class are visible in the derived class
- A class declared as `final` cannot be inherited

Method Overriding

- If a method in a subclass has identical name and signature to some method in its superclass, then the method definition in the subclass *overrides* the method definition provided in the superclass
- Note that, if only method names are identical but the signatures are different, then method overloading happens, and both of them stays visible in derived class
- The older(superclass) definition gets suppressed by the newer(subclass) definition - normal invocations will execute the newer definition!

Method Overriding

- If a method in a subclass has identical name and signature to some method in its superclass, then the method definition in the subclass *overrides* the method definition provided in the superclass
- Note that, if only method names are identical but the signatures are different, then method overloading happens, and both of them stays visible in derived class
- The older(superclass) definition gets suppressed by the newer(subclass) definition - normal invocations will execute the newer definition!
- Data members can also be suppressed/overridden similarly

Method Overriding

- If a method in a subclass has identical name and signature to some method in its superclass, then the method definition in the subclass *overrides* the method definition provided in the superclass
- Note that, if only method names are identical but the signatures are different, then method overloading happens, and both of them stays visible in derived class
- The older(superclass) definition gets suppressed by the newer(subclass) definition - normal invocations will execute the newer definition!
- Data members can also be suppressed/overridden similarly
- Question: can we access base class version of a method?

`super` Power!

- The `super` is used to refer to the immediate superclass of the current object
- `super` is used analogous to `this` keyword

`super` Power!

- The `super` is used to refer to the immediate superclass of the current object
- `super` is used analogous to `this` keyword
- By writing `super.member` we can access any member (both data & methods) available in the superclass
- This is useful to access overridden members of the superclass from subclass

super Power!

- The `super` is used to refer to the immediate superclass of the current object
- `super` is used analogous to `this` keyword
- By writing `super.member` we can access any member (both data & methods) available in the superclass
- This is useful to access overridden members of the superclass from subclass
- We can invoke a constructor of the immediate superclass by calling `super(args)`
- The call to the superclass constructor must always be the first statement executed inside a subclass constructor
- When we don't explicitly write a `super(args)` statement, Java implicitly calls the default parameterless constructor of the parentclass
- In case of multilevel hierarchy the execution of constructor calls is same as order of derivation

Dynamic Method Dispatch

- A superclass variable can be assigned reference of a subclass object

```
BaseClass obj = new DerivedClass();
```

Dynamic Method Dispatch

- A superclass variable can be assigned reference of a subclass object

```
BaseClass obj = new DerivedClass();
```

- We can now access visible `BaseClass` members which were only inherited into `DerivedClass` though `obj`
- The new members introduced `DerivedClass` cannot be directly accessed without an explicit typecasting

Dynamic Method Dispatch

- A superclass variable can be assigned reference of a subclass object

```
BaseClass obj = new DerivedClass();
```

- We can now access visible `BaseClass` members which were only inherited into `DerivedClass` though `obj`
- The new members introduced `DerivedClass` cannot be directly accessed without an explicit typecasting
- In case of overridden members, the derived class version gets attended

Dynamic Method Dispatch

- A superclass variable can be assigned reference of a subclass object

```
BaseClass obj = new DerivedClass();
```

- We can now access visible `BaseClass` members which were only inherited into `DerivedClass` though `obj`
- The new members introduced `DerivedClass` cannot be directly accessed without an explicit typecasting
- In case of overridden members, the derived class version gets attended
- We can call an overridden method though a base class variable holding reference of a derived class object

To be continued...