Virtual Method-

*In object-oriented programming, a virtual function or virtual method is a function or method whose behaviour can be overridden within an inheriting class by a function with the same signature to provide the polymorphic behavior.*

Therefore according to definition, every **non-static method** in JAVA is by default *virtual method* **except final and private methods**. The methods which cannot be inherited for polymorphic behavior is not a virtual method.

|  |  |  |
| --- | --- | --- |
| 1 | import java.util.\*; | |
| 2 |  |

|  |  |  |
| --- | --- | --- |
| 3 | public class Animal | |
| 4 | { |

|  |  |  |
| --- | --- | --- |
| 5 | public void eat() | |
| 6 | { |

|  |  |  |
| --- | --- | --- |
| 7 | System.out.println("I eat like a generic Animal."); | |
| 8 | } |

|  |  |  |
| --- | --- | --- |
| 9 |  | |
| 10 | | public static void main(String[] args) | |

|  |  |
| --- | --- |
| 11 | { |
| 12 | List<Animal> animals = new LinkedList<Animal>(); | |

|  |  |
| --- | --- |
| 13 |  |
| 14 | animals.add(new Animal()); | |

|  |  |
| --- | --- |
| 15 | animals.add(new Wolf()); |
| 16 | animals.add(new Fish()); |

|  |  |
| --- | --- |
| 17 | animals.add(new Goldfish()); |
| 18 | animals.add(new OtherAnimal()); | |

|  |  |
| --- | --- |
| 19 |  |
| 20 | for (Animal currentAnimal : animals) | |

|  |  |
| --- | --- |
| 21 | { |
| 22 | currentAnimal.eat(); | |

|  |  |  |
| --- | --- | --- |
| 23 | } | |
| 24 | } |

|  |  |  |
| --- | --- | --- |
| 25 | } | |
| 26 |  |

|  |  |  |
| --- | --- | --- |
| 27 | public class Wolf extends Animal | |
| 28 | { |

|  |  |
| --- | --- |
| 29 | @Override |
| 30 | public void eat() | |

|  |  |
| --- | --- |
| 31 | { |
| 32 | System.out.println("I eat like a wolf!"); | |

|  |  |  |
| --- | --- | --- |
| 33 | } | |
| 34 | } |

|  |  |
| --- | --- |
| 35 |  |
| 36 | public class Fish extends Animal | |

|  |  |
| --- | --- |
| 37 | { |
| 38 | @Override | |

|  |  |  |
| --- | --- | --- |
| 39 | public void eat() | |
| 40 | { |

|  |  |  |
| --- | --- | --- |
| 41 | System.out.println("I eat like a fish!"); | |
| 42 | } |

|  |  |  |
| --- | --- | --- |
| 43 | } | |
| 44 |  |

|  |  |  |
| --- | --- | --- |
| 45 | public class Goldfish extends Fish | |
| 46 | { |

|  |  |
| --- | --- |
| 47 | @Override |
| 48 | public void eat() | |

|  |  |
| --- | --- |
| 49 | { |
| 50 | System.out.println("I eat like a goldfish!"); | |

|  |  |  |
| --- | --- | --- |
| 51 | } | |
| 52 | } |

|  |  |
| --- | --- |
| 53 |  |
| 54 | public class OtherAnimal extends Animal {} | |

Output:

*I eat like a generic Animal.  
I eat like a wolf!  
I eat like a fish!  
I eat like a goldfish!  
I eat like a generic Animal.*

**Abstract class is nothing but the pure virtual method equivalent to C++ in Java.**

**Question : why we say that static method is not a virtual method in Java?**

Answer : static method is bound to the class itself, so calling the static method from class name or object does not provide the polymorphic behavior to the static method. We can override the static method however it will not give the advantage of the polymorphism