**- INTERFACE –**

|  |
| --- |
| **Interface:**   * A basic blueprint to present a standard with no actual implementation (Implementation is done in the Implemented class). * Enables polymorphism: One object can be made into many different types and can be changed as desired by the developer, just like parts of a product. * All member variables must be **public static final**, and static final can be omitted.'\ * Interfaces cannot create objects. However, it is used only as a variable type.   **public interface InterfaceEx {**  **public /\*static final\*/ int CONSTANT\_NUM = 100;**  **public /\*abstract\*/ void method1();** |

1. **OVERLOADING & OVERRIDING**

Conditions for overriding:

1. The declaration part must be the same (name, parameter, return type)
2. The access controller cannot be changed to a narrow range (e.g. If an ancestor class method is protected, it can only be set to protected or public with the same or wider scope)

**Overloading:** Defining a new method that does not exist in the compiler's point of view **(multi-defining** a method - the same method in the same class has multiple parameters with different parameters)

**Overriding:** Changing the contents of an inherited method (just bring the template and **redefining a method** - The same method exists in parent class and child class

1. **TYPE OF METHODS** 
   1. **CONSTANT METHOD**

* “Static final” doesn’t have to be written (complier automatically adds it)

**public interface InterfaceEx1 {**

**public static final int MIN\_PRICE = 0;**

**public int MAX\_PRICE = 1000;}** //static final omitted

* 1. **ABSTRACT METHOD**
* “abstract” doesn’t have to be written (complier automatically adds it)

**public interface InterfaceEx2 {**

**public abstract double meanPrice();**

**public double totalPrice();**

* 1. **DEFUALT METHOD**
* The keyword “default” must be written
* A method with an implementation (execution block)

**public interface InterfaceEx3 {**

**default double getSalePrice(double price) {**

**return price – (price\*0.05);**

* 1. **STATIC METHOD**
* A method with an implementation (execution block)
* Even without a object, the method can be called with interface alone

**public interface InterfaceEx3 {**

**Static void printPrice(double price) {**

**System.out.println(price);**

1. **DOWNCASTING**

|  |  |
| --- | --- |
| EXAMPLE | |
| **Interface** |  |
| **class** |  |
| **TestMain** |  |
| **Result** | **Sleeping**  **Eating** |

1. **INSTANCEOF**

|  |  |  |
| --- | --- | --- |
| **Instance of** | Checking which class the object is from, or inherited from | **Object instanceOf type**  **Example:**  **Public class Banana implements fruit {**  **}**  **Public apple {**  **}**  **Main class:**  **Banana fruit = new Banana ();**  **System.out.println(fruit instancof Banana);**  **System.out.println(fruit instancof Apple);**  **Result:**  **True**  **True** |