Day -7

Abstract class

Abstract and Extends key words

Rules of abstract class

Inheritance

Multilevel inheritance

Hybrid inheritance

Interfaces

* It’s a collection variables but each variable is public static final
* Cannot have a constructor
* Interface cannot be Instantiated
* Pure abstract class

Why Interfaces

* Since java doesn’t supports the multi-level inheritance, to achieve the multi-level inheritance Interfaces are used.

Syntax:

Interface <InterfaceName>{

Public static final int VAR = value;

Public void display(); //method declaration (No body to the method)

}

How to use?

ClassName implements InterfaceName{}

ClassName implements InterfaceName1, InterfaceName2{}

Static Keyword

Static is a keyword or reserved word in the java.

Memory is pre created.

Only one copy is created.

* + Static Variable
    - Ex: static int intVar
  + Static method
    - Ex: static int add(){return 10};
  + Static block
    - Static{}

Static can be called as class level variable. Whereas Instance variables need to instantiate first and need to be called.

Variables Types

* **Instance Variables (Non-Static Fields)** Technically speaking, objects store their individual states in "non-static fields", that is, fields declared without the static keyword. Non-static fields are also known as *instance variables* because their values are unique to each *instance* of a class (to each object, in other words); the currentSpeed of one bicycle is independent from the currentSpeed of another.
* **Class Variables (Static Fields)** A *class variable* is any field declared with the static modifier; this tells the compiler that there is exactly one copy of this variable in existence, regardless of how many times the class has been instantiated. A field defining the number of gears for a particular kind of bicycle could be marked as static since conceptually the same number of gears will apply to all instances. The code static int numGears = 6; would create such a static field. Additionally, the keyword final could be added to indicate that the number of gears will never change.
* **Local Variables** Similar to how an object stores its state in fields, a method will often store its temporary state in *local variables*. The syntax for declaring a local variable is similar to declaring a field (for example, int count = 0;). There is no special keyword designating a variable as local; that determination comes entirely from the location in which the variable is declared — which is between the opening and closing braces of a method. As such, local variables are only visible to the methods in which they are declared; they are not accessible from the rest of the class.
* **Parameters** You've already seen examples of parameters, both in the Bicycle class and in the main method of the "Hello World!" application. Recall that the signature for the main method is public static void main(String[] args). Here, the args variable is the parameter to this method. The important thing to remember is that parameters are always classified as "variables" not "fields". This applies to other parameter-accepting constructs as well (such as constructors and exception handlers) that you'll learn about later in the tutorial.