Object-Oriented Programming –A Brief Overview
Lecture-3

Some Java keywords along with brief descriptions of their functions:

abstract

The abstract keyword is used to declare a class or method to be abstract. An abstract method has no implementation; all classes containing abstract methods must themselves be abstract, although not all abstract classes have abstract methods. Objects of a class which is abstract cannot be instantiated, but can be extended by other classes. All subclasses of an abstract class must either provide implementations for all abstract methods, or must also be abstract.

boolean

The boolean keyword is used to declare a field that can store a boolean value; that is, either true or false. This keyword is also used to declare that a method returns a value of type boolean.

break

Used to resume program execution at the statement immediately following the current enclosing block or statement. If followed by a label, the program resumes execution at the statement immediately following the enclosing labeled statement or block.

byte

The byte keyword is used to declare a field that can store an 8-bit signed two's complement integer. This keyword is also used to declare that a method returns a value of type byte.

case

The case keyword is used to create individual cases in a switch statement; see *switch*.

catch

Defines an exception handler—a group of statements that are executed if an exception is thrown in the block defined by a preceding try keyword. The code is executed only if the class of the thrown exception is assignment compatible with the exception class declared by the catch clause.

char

The char keyword is used to declare a field that can store a 16-bit Unicode character. This keyword is also used to declare that a method returns a value of type char.

class

A type that defines the implementation of a particular kind of object. A class definition defines instance and class fields, methods, and inner classes as well as specifying the interfaces the class implements and the immediate superclass of the class. If the superclass is not explicitly specified, the superclass is implicitly Object.

const

Although reserved as a keyword in Java, const is not used and has no function.

continue

Used to resume program execution at the end of the current loop body. If followed by a label, continue resumes execution at the end of the enclosing labeled loop body.

default

The default can optionally be used in a switch statement to label a block of statements to be executed if no case matches the specified value.

do

The do keyword is used in conjunction with while to create a do-while loop, which executes a block of statements associated with the loop and then tests a boolean expression associated with the while. If the expression evaluates to true, the block is executed again; this continues until the expression evaluates to false.

double

The double keyword is used to declare a field that can hold a 64-bit double precision floating-point number. This keyword is also used to declare that a method returns a value of type double.

else

The else keyword is used in conjunction with if to create an if-else statement, which tests a boolean expression; if the expression evaluates to true, the block of statements associated with the if are evaluated; if it evaluates to false, the block of statements associated with the else are evaluated.

extends

Used in a class declaration to specify the superclass; used in an interface declaration to specify one or more superinterfaces. Class X extends class Y to add functionality, either by adding fields or methods to class Y, or by overriding methods of class Y. An interface Z extends one or more interfaces by adding methods. Class X is said to be a subclass of class Y; Interface Z is said to be a subinterface of the interfaces it extends.

final

Define an entity once that cannot be changed nor derived from later. More specifically: a final class cannot be subclassed, a final method cannot be overridden, and a final variable can occur at most once as a left-hand expression. All methods in a final class are implicitly final.

Example of Polymorphism

```
package Polymorphism;

public class Animal
{
   public String talk()
   {
      return "Can Animal Talk!";
   }
}

package Polymorphism;

class Cat extends Animal
{
   public String talk()
   {
      return "Meow!";
   }
}

package Polymorphism;

class Dog extends Animal
{
      public String talk()
      {
         return "Woof!";
      }
}

package Polymorphism;

class Cow extends Animal
{
      public String talk()
      {
         return "Hamba!";
      }
}

public class TestPolymorphism
{
      public static void main(String[] args)
      {
         Cat cat = new Cat();
         Dog dog = new Dog();
           Cow cow = new Cow ();

         System.out.println("Cat Do: " + cat.talk());
         System.out.println("Dog Do: " +dog.talk());
           System.out.println("Cow Do: " +cow.talk());
      }
}
```