## **Reading complex pointers**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Precedence** | **Associativity** |
| (), [] | 1 | Left to right |
| \*, identifier | 2 | Right to left |
| Data type | 3 | - |

**1.How to read the pointer: int (\*p)[10].**

->p is a pointer to an array of integers of size 10.

**2.int** (\*p)(**int** (\*)[2], **int** (\*)**void**))

p is a pointer to such function which accepts the first parameter as the pointer to a one-dimensional array of integers of size two and the second parameter as the pointer to a function which parameter is void and return type is the integer.

**Print the Address of variable or pointer**

when you want to print a character variable use %c, an int , use %d, a string , use %s,  **to print address as stored in pointer use %p .**

When you use %p as in

printf("%p\n",ptr);

%p is the format specifier for pointers, you get the expected results

When you use %d as in

printf("%d\n",ptr);

%d is the format specifier for signed integers, you get unexpected results because the size of pointers (which is usually 8 bytes) can be different from size of signed integers(which is usually 4 bytes).

**It is not possible to get an address of any primitive variable or object in Java.**

## **Types Of Pointers In C Programming**

* **NULL Pointer**
* **Dangling Pointer**
* **Generic Pointers(void pointer)**
* **Wild Pointer**
* **Complex Pointers**
* **Near Pointer**
* **Far Pointer**
* **Huge Pointers**

**Static variable:-**

The **static** variable maintains **single copy** for a **whole** **class**. The **static** variables cannot be declared within a method or block, bz it is class level not local. If the value of a **static** variable changed **by** an object then it gets reflected into all the objects of the **class**.

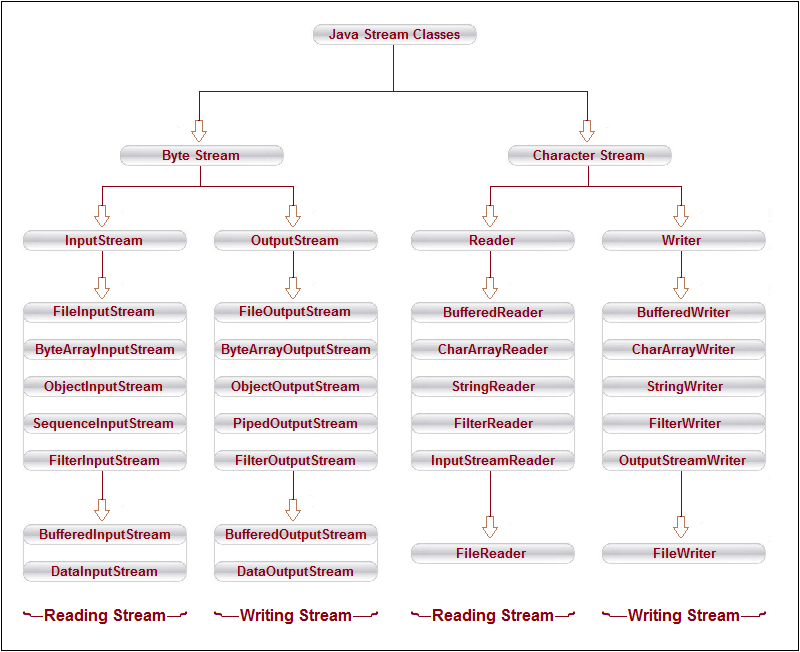
**Static Method:-**

**Static methods can not** inherited then we can say it can not be **overridden** because they are not part of the object's state. Rather, they belongs to the class (i.e they are class **methods**).

1. Instance method are methods which **require an object** of its class to be created before it can be called. Static methods are the methods in Java that can be called **without creating an object** of class.
2. Static method means which will exist as a **single copy for a whole class**. But instance methods exist as multiple copies depending on the number of instances created for that class.
3. Static methods can’t access instance methods and instance variables **directly**. Instance method can access static variables and static methods directly.

**Reader/Writer and InputStream/OutputStream**

The only difference is that the methods in the **Writer** and **Reader** classes are designed to deal with **character input/output**, while the corresponding methods in the **OutputStream** and **InputStream** classes are designed to deal with **byte input/output.**



# **Generalization**

Generalization is like a bottom-up approach. It is process of extracting the shared characteristics from two or more classes and combining them into generalized super class.

Example:- **Abstract Class**

# **Specialization**

Specialization is a top-down approach, creating a new classes from existing class.

Example :- **Inheritance**

**Access Specifier:**

It is a like some **restrictions** on the class members not to get directly accessed by the outside functions.

There are 3 types of access modifiers available in C++:

In C++, default the access modifier for the members will be **Private**

1. **Public**
2. **Private**
3. **Protected**

In java**,** If you don't use any modifier, it is treated as **default** by default. The default modifier is accessible only within package.

There are 4 types of java access modifiers:

1. private
2. default
3. protected
4. public

|  |  |  |  |
| --- | --- | --- | --- |
| **Access** | **public** | **protected** | **private** |
| Same class | yes | yes | yes |
| Derived classes | yes | yes | no |
| Outside classes | yes | no | no |

**The protected and private access modifier can be applied on the data member, method and constructor.** **It can't be applied on the class.(only use it in nested class).**

|  |
| --- |
| A class cannot be private or protected except nested class. **If you make any class constructor private, you cannot create the instance of that**  **class from outside the class.(we need to create in same class).**  **For example:** |

**class** A{

**private** A(){}//private constructor

**void** msg()

{

System.out.println("Hello java");}

}

**public** **class** Simple{

**public** **static** **void** main(String args[])

{

    A obj=**new** A();//Compile Time Error

  }

If you don't use any modifier, it is treated as **default** by default. The default modifier is accessible only within package.

**package** pack;

**class** A{

**void** msg(){System.out.println("Hello");}

}

//save by B.java

**package** mypack;

**import** pack.\*;

**class** B{

**public** **static** **void** main(String args[]){

    A obj = **new** A();//Compile Time Error

    obj.msg();//Compile Time Error

   }

}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Access Modifier** | **within class** | **within package** | **outside package by subclass only** | **outside package** |
| **Private** | Y | N | N | N |
| **Default** | Y | Y | N | N |
| **Protected** | Y | Y | Y | N |
| **Public** | Y | Y | Y | Y |

|  |
| --- |
|  |

**Important points about overriding.**

* If the overridden is **public**, then the overriding method must be **public**; otherwise, a compile-time error occurs.
* If the overridden method is **protected,** then the overriding method must be **protected or public**; otherwise, a compile-time error occurs.
* If the overridden method has **default** (package) access, then the overriding method **must not** be **private**; otherwise, a compile-time error occurs.
* **If the overridden method has Private (package) access, we cant override it.**

**Above overriding rule same for abstract method also.**

**Abstract method can be public/protected/default but not private.because private not inherited.** The derive class cannot access the private method (even using ***super***).

**Override Private Method:-**

You can't override a private method, but you can introduce one in a derived class without a problem. The derive class cannot access the private method (even using ***super***)

class A{

private void msg()

{

System.out.println("Hello Base");

}

}

public class MyClass extends A

{

void print()

{

super.msg();// **Error**

System.out.println("Hello Derived");

}

public static void main(String args[])

{

MyClass obj=new MyClass();

obj.print();

}

* A virtual function must be defined in the base class, even though it is not used.
* Virtual functions cannot be static members.

### **What are the advantages of multithreading?**

1. better utilization of cache memory as threads share the common memory resources.

2. one server can execute multiple threads at a time.

### **What is context switching?**

In Context switching the state of the process (or thread) is stored so that it can be restored and execution can be start from the same point later. Context switching enables the multiple processes to share the same CPU.

**Points related to NetBeans:**

**Java 10.0.1 in Netbeans cant run ,so we need to change JDK path ,so go->**

**1. C:\Program Files\NetBeans 8.2\etc.**

**2.open file “netbeans.conf” change “netbeans\_jdkhome="C:\Program Files\Java\jdk10.0.1" To “netbeans\_jdkhome="C:\Program Files\Java\jdk1.8.0\_131".**

**3. Start Neatbeans .**

**How to start MySql on Cmd**

**Before going to following step we need to start local server**

**1. open cmd**

**2. write cd<space>C:\xampp\mysql\bin**

**3. Enter**

**4. now write after the C:\xampp\mysql\bin>**

**5. mysql.exe –h localhost –u root –p and Enter**

**6.will Show “Enter Password”(Don’t enter password”) and press Enter directly**

**7.type show databases; OR directly use database by typing “use db1”.**

**General Command after start MySql**

**1.Show databases;**

**2.Use database\_name;**

**3.Show tables;**

**4.show *columns* from table\_name;(or show *columns* from database\_name.table\_name;)**