**Which Catch Block to Execute(REF.HeadFirstJava Pno:333)**

* The JVM simply starts at the first one and works its way down until it finds a catch that's broad enough (in other words, high enough on the inheritance tree) to handle the exception.
* if your first catch block is catch (Exception ex). the compiler knows there's no point in adding any others, and they'll never be reached.

**Difference Between Exception and error**

|  |  |
| --- | --- |
| **Error** | **Exception** |
| It is a class  An error is irrecoverable.  Errors can occur at compile time as well as run time.: eg Syntax Error  As the error is detected the program will terminated abnormally.  Errors takes more time to recover  . | It is a object  An Exception is recoverable.  Exception Occurs At RunTime  As an exception is detected, if it is handled program is not Terminated  abnormally  Exceptions take less time to recover |

**Finally Block**

* When we write System.exit() in try block

**NullPointerException(Ref Oracle docs)**

* Calling the instance method of a null object.
* Accessing or modifying the field of a null object.
* Taking the length of null as if it were an array.
* Accessing or modifying the slots of null as if it were an array.
* Throwing null as if it were a Throwable value**.**

**Constructor Can Be Private And Protected(ref headfirstjava pfno:249)**

* Constructors can be public, private, protected,or default
* If we declare constructor as private and protected we can not perform constructor chaining.

**Multiple inheritance with Interface**

* A java *interface* solves your multiple inheritance problem by giving you much of the polymorphic benefits of multiple inheritance without the pain and suffering from deadly diamond deadly.

**When To Use Interface And Abstract(**ref headfirstjava pgno:227)

* Use an abstract class when you want to define a template for a group of subclasses, and you have at least some implementation code that all subclasses could use.
* Make the class abstract when you want to guarantee that nobody can make object type.
* Use an interface when you want to define a role that other classes can play, regardless of where those classes are in inheritance tree

**Class Can Extend only One Class**

* A java class can have only one parent class and that parent class defines who you are. But if a class has two parent class it creates diamond problem.(which method to execute constructor to call)
* Interface there are no instance variables, thus no constructor, methods are abstract so no diamond problem.