BEA WebLogic Workshop™ Help

Version 8.1 SP4
December 2004

Table of Contents

Table of Contents

Table of Contents

Java Language Keywords

The Java language defines the following keywords and reserved words.

Topics Included in This Section

The instanceof Keyword

The int Keyword

The interface Keyword

The long Keyword

The new Keyword

The null Keyword

The package Keyword

The private Keyword

The protected Keyword

The public Keyword

The return Keyword

The short Keyword

The static Keyword

The super Keyword

The switch Keyword

The synchronized Keyword

The this Keyword

The throw Keyword

The throws Keyword

The transient Keyword

The try Keyword

The true Keyword

The void Keyword

The while Keyword

Related Topics

None.

abstract Java Keyword

The abstract keyword may modify a class or a method.

An abstract class can be extended (subclassed) but cannot be instantiated directly.

An abstract method is not implemented in the class in which it is declared, but must be overridden in some subclass.

Examples

```
public abstract class MyClass
{
}


public abstract String myMethod();
```

Remarks

- A class with an abstract method is inherently abstract and must be declared abstract.
- An abstract class cannot be instantiated.
- A subclass of an abstract class can only be instantiated if it implements all of the abstract methods of its superclass. Such classes are called concrete classes to differentiate them from abstract classes.
- If a subclass of an abstract class does not implement all of the abstract methods of its superclass, the subclass is also abstract.
- The abstract keyword cannot be applied to methods that are static, private or final, since such methods cannot be overridden and therefore cannot be implemented in subclasses.
- No methods of a final class may be abstract since a final class cannot be subclassed.

Related Topics

final Java Keyword

private Java Keyword

static Java Keyword

boolean Java Keyword

boolean is a Java primitive type.

A boolean variable may take on one of the values true or false.

Examples

```
boolean valid = true;


if (valid)
{
   <statement>
}
```

Remarks

- A boolean variable may only take on the values true or false. A boolean may not be converted to or from any numeric type.
- Expressions containing boolean operands can contain only boolean operands.
- The Boolean class is a wrapper class for the boolean primitive type.

Related Topics

false Java Keyword

true Java Keyword

break Java Keyword

The break keyword is used to prematurely exit a for, while, or do loop or to mark the end of a case block in a switch statement.

Examples

```
for (i=0; i<max; i++)
{
   if (<loop finished early>)
   {
      break;
   }
}


int type = <some value>;
switch (type)
{
   case 1:
         <statement>
 break;
   case 2:
         <statement>
 break;
   default:
         <statement>
}
```

Remarks

- break always exits the innermost enclosing while, for, do or switch statement.

Related Topics

continue Java Keyword

do Java Keyword

for Java Keyword

switch Java Keyword

while Java Keyword

byte Java Keyword

byte is a Java primitive type.

A byte can store an integer value in the range [−128, 127].

Examples

```
  byte b = 124;
```

Remarks

- The Byte class is a wrapper class for the byte primitive type. It defines MIN_VALUE and MAX_VALUE constants representing the range of values for this type.
- All integer literals in Java are 32−bit int values unless the value is followed by l or L as in 235L, indicating the value should be interpreted as a long.

Related Topics

int Java Keyword

long Java Keyword

short Java Keyword

case Java Keyword

The case is used to label each branch in a switch statement.

Examples

```
int arg = <some value>;
switch (arg)
{
   case 1:
      <statements>
      break;
   case 2:
      <statements>
      break;
   default:
      <statements>
      break;
}
```

Remarks

- A case block does not have an implicit ending point. A break statement is typically used at the end of each case block to exit the switch statement.
- Without a break statement, the flow of execution will flow into all following case and/or default blocks.

Related Topics

break Java Keyword

default Java Keyword

switch Java Keyword

catch Java Keyword

The catch keyword is used to define exception handling blocks in try−catch or try−catch−finally statements.

Examples

```
try
{
   <block that may throw exceptions>
}
catch (<java.lang.Exception or subclass> e)
{
   <code to handle exception e>
}
```

```
try
{
   <block that may throw different exceptions>
}
catch (FooException e)
{
   <code to handle FooException e>
}
catch (BarException e)
{
   <code to handle BarException e>
}
```

```
try
{
   <block that may throw exceptions>
}
catch (<java.lang.Exception or subclass> e)
{
   <code to handle exception e>
}
finally
{
   <statements that execute with or without exception>
}
```

Remarks

- The opening and closing curly braces { and } are part of the syntax of the catch clause and may not be omitted even if the clause contains a single statement.
- Every try block must have at least one catch or finally clause.
- If a particular exception class is not handled by any catch clause, the exception propagates up the call stack to the next enclosing try block, recursively. If an exception is not caught by any enclosing try block, the Java interpretor will exit with an error message and stack trace.

Related Topics

finally Java Keyword

try Java Keyword

char Java Keyword

char is a Java primitive type.

A char variable can store a single Unicode character.

Examples

```
char delimiter = ';';
```

Remarks

- The following char constants are available:
    - \b − Backspace
    - \f − Form feed
    - \n − Newline
    - \r − Carriage return
    - \t − Horizontal tab
    - \' − Single quote
    - \" − Double quote
    - \" − Backslash
    - \xxx − The Latin−1 character with the encoding xxx. The \x and \xx forms are legal but may lead to confusion.
    - \uxxxx − The Unicode character with the hexadecimal encoding xxxx.
- The Character class includes useful static methods for dealing with char variables, including isDigit(), isLetter(), isWhitespace() and toUpperCase().
- char values are unsigned.

Related Topics

None

class Java Keyword

The class keyword is used to declare a new Java class, which is a collection of related variables and/or methods.

Classes are the basic building blocks of object−oriented programming. A class typically represents some real−world entity such as a geometric Shape or a Person. A class is a template for an object. Every object is an *instance* of a class.

To use a class, you instantiate an object of the class, typically with the new operator, then call the classes methods to access the features of the class.

Examples

```
public class Rectangle
{
   float width;
   float height;

   public Rectangle(float w, float h)
   {
     width = w;
     height = h;
   }

   public float getWidth()
   {
      return width;
   }

   public float getHeight()
   {
      return height;
   }
}
```

Remarks

None.

Related Topics

new Java Keyword

continue Java Keyword

The continue keyword is used to skip to the next iteration of a for, while, or do loop.

Examples

```
for (i=0; i<max; i++)
{
   <statements>
   if (<done with this iteration>)
   {
      continue;
   }
   <statements>
}
```

Remarks

- continue always skips to the next iteration of the innermost enclosing while, for or do statement.

Related Topics

break Java Keyword

do Java Keyword

for Java Keyword

while Java Keyword

default Java Keyword

The default keyword is used to label the default branch in a switch statement.

Examples

```
int arg = <some value>;
switch (arg)
{
   case 1:
      <statements>
      break;
   case 2:
      <statements>
      break;
   default:
      <statements>
      break;
}
```

Remarks

- A default block does not have an implicit ending point. A break statement is typically used at the end of each case or default block to exit the switch statement upon completion of the block.
- Without a default statement, a switch statement whose argument matches no case blocks will do nothing.

Related Topics

break Java Keyword

case Java Keyword

switch Java Keyword

do Java Keyword

The do keyword specifies a loop whose condition is checked at the *end* of each iteration.

Examples

```
do
{
   <statements>
}
while (!found);
```

Remarks

- The body of a do loop is always executed at least once.
- The semicolon after the condition expression is always required.

Related Topics

break Java Keyword

continue Java Keyword

for Java Keyword

while Java Keyword

double Java Keyword

double is a Java primitive type.

A double variable may store a double−precision floating point value.

Examples

```
double ratio = .01;
double diameter = 6.15;
double height = 1.35E03;  // 1.35 * $10^{3}$ or 1350.0
double height = 1e−2;  // 1.0 * $10^{-2}$ or 0.01
```

Remarks

- Since floating point data types are approximations of real numbers, you should generally never compare floating point numbers for equality.
- Java floating point numbers can represent *infinity* and *NaN* (not a number). The Double wrapper class defines the constants MIN_VALUE, MAX_VALUE, NEGATIVE_INFINITY, POSITIVE_INFINITY and NaN.

Related Topics

float Java Keyword

else Java Keyword

The else keyword is always used in association with the if keyword in an if−else statement. The else clause is optional and is executed if the if condition is false.

Examples

```
if (condition)
{
   <statements>
}
else
{
   <statements>
}
```

Remarks

None.

Related Topics

if Java Keyword

extends Java Keyword

The extends keyword is used in a class or interface declaration to indicate that the class or interface being declared is a subclass of the class or interface whose name follows the extends keyword.

Examples

```
  public class Rectangle extends Polygon
  {
  }
```

Remarks

- In the example above, the Rectangle class inherits all of the public and protected variables and methods of the Polygon class.
- The Rectangle class may override any non−final method of the Polygon class.
- A class may only extend one other class.

Related Topics

class Java Keyword

implements Java Keyword

interface Java Keyword

false Java Keyword

The false keyword represents one of the two legal values for a boolean variable.

Examples

```
  boolean isComplete = false;
```

Remarks

None.

Related Topics

boolean Java Keyword

true Java Keyword

final Java Keyword

The final keyword may be applied to a class, indicating the class may not be extended (subclassed).

The final keyword may be applied to a method, indicating the method may not be overridden in any subclass.

Examples

```
public final class MyFinalClass
{
}


public class MyClass
{
   public final String myFinalMethod()
       {
          <statements>
       }
}
```

Remarks

- A class may never be both abstract and final. abstract means the class must be extended, while final means it cannot be.
- A method may never be both abstract and final. abstract means the method must be overridden, while final means it cannot be.

Related Topics

abstract Java Keyword

finally Java Keyword

The finally keyword is used to define a block that is *always* executed in a try−catch−finally statement.

A finally block typically contains cleanup code that recovers from partial execution of a try block.

Examples

```
try
{
   <block that may throw exceptions>
}
catch (<java.lang.Exception or subclass> e)
{
   <code to handle exception e>
}
finally
{
   <statements that execute with or without exception>
}
```

Remarks

- The opening and closing curly braces { and } are part of the syntax of the finally clause and may not be omitted even if the clause contains a single statement.
- Every try block must have at least one catch or finally clause.
- If any portion of the try block is executed, the code in a finally block is always guaranteed to be executed whether an exception occurs or not and independent of whether the try or catch blocks contain return, continue or break statements.
- In the absence of exceptions, control flows through the try block and then into the finally block.
- If an exception occurs during execution of the try block and the appropriate catch block contains a break, continue or return statement, control flows through the finally block before the break, continue or return occurs.

Related Topics

finally Java Keyword

try Java Keyword

float Java Keyword

float is a Java primitive type.

A float variable may store a single−precision floating point value.

Examples

```
float ratio = .01;
float diameter = 6.15;
float height = 1.35E03;  // 1.35 * 10^3 or 1350.0
float height = 1e−2;  // 1.0 * 10^-2 or 0.01
```

Remarks

The following rules apply to this keyword's use:

- Floating point literals in Java always default to double−precision. To specify a single−precision literal value, follow the number with f or F, as in 0.01f.
- Since floating point data types are approximations of real numbers, you should generally never compare floating point numbers for equality.
- Java floating point numbers can represent *infinity* and *NaN* (not a number). The Float wrapper class defines the constants MIN_VALUE, MAX_VALUE, NEGATIVE_INFINITY, POSITIVE_INFINITY and NaN.

Related Topics

double Java Keyword

for Java Keyword

The for keyword specifies a loop whose condition is checked before each iteration.

Examples

```
int i;
for (i=0; i<max; i++)
{
   <statements>
}
```

Remarks

- The for statement takes the form for(*initialize*; *condition*; *increment*)

  The *initialize* statement is executed once as the flow of control enters the for statement.

  The *condition* is evaluated before each execution of the body of the loop. The body of the loop is executed if the *condition* is true.

  The *increment* statement is executed after each execution of the body of the loop, before the *condition* is evaluated for the next iteration.

Related Topics

break Java Keyword

continue Java Keyword

do Java Keyword

while Java Keyword

if Java Keyword

The if keyword indicates conditional execution of a block. The condition must evaluate to a boolean value.

Examples

```
if (condition)
{
   <statements>
}


if (condition)
{
   <statements>
}
else
{
   <statements>
}
```

Remarks

- An if statement may have an optional else clause containing code that is executed if the condition is false.
- Expressions containing boolean operands can contain only boolean operands.

Related Topics

boolean Java Keyword

else Java Keyword

implements Java Keyword

The implements keyword is used in a class declaration to indicate that the class being declared provides implementations for all methods declared in the interface whose name follows the implements keyword.

Examples

```
public class Truck implements IVehicle
{
}
```

Remarks

- In the example above, the Truck class must provide implementations for all methods declared in the IVehicle interface.
- The Truck class is otherwise independent; it may declare additional methods and variables and may extend another class.
- A single class may implement multiple interfaces.

Related Topics

class Java Keyword

extends Java Keyword

interface Java Keyword

import Java Keyword

The import keyword makes one class or all classes in a package visible in the current Java source file. Imported classes can be referened without the use of fully−qualified class names.

Examples

```
  import java.io.File;
  import java.net.*;
```

Remarks

- Many Java programmers use only specific import statements (no '*') to avoid ambiguity when multiple packages contain classes of the same name.

Related Topics

package Java Keyword

instanceof Java Keyword

The instanceof keyword is used to determine the class of an object.

Examples

```
if (node instanceof TreeNode)
{
   <statements>
}
```

Remarks

- In the example above, if node is an instance of the TreeNode class or is an instance of a subclass of TreeNode, the instanceof expression evaluates to true.

Related Topics

None.

int Java Keyword

int is a Java primitive type.

A int variable may store a 32−bit integer value.

Examples

```
int number = 5;
int octalNumber = 0377;
int hexNumber = 0xff;
```

Remarks

- The Integer class is a wrapper class for the int primitive type. It defines MIN_VALUE and MAX_VALUE constants representing the range of values for this type.
- All integer literals in Java are 32−bit int values unless the value is followed by l or L as in 235L, indicating the value should be interpreted as a long.

Related Topics

byte Java Keyword

long Java Keyword

short Java Keyword

interface Java Keyword

The interface keyword is used to declare a new Java interface, which is a collection of methods.

Interfaces are a powerful feature of the Java language. Any class may declare that it implements one or more interfaces, meaining it implements all of the methods defined in those interfaces.

Examples

```
public interface IPolygon
{
   public float getArea();
   public int getNumberOfSides();
   public int getCircumference();
}
```

Remarks

- Any class that implements an interface must provide implementations for all methods in that interface.
- A single class may implement multiple interfaces.

Related Topics

class Java Keyword

long Java Keyword

long is a Java primitive type.

A long variable may store a 64−bit signed integer.

Examples

```
long number = 5;
long anotherNumber = 34590L;
long octalNumber = 0377;
long hexNumber = 0xffl;
```

Remarks

- The Long class is a wrapper class for the long primitive type. It defines MIN_VALUE and MAX_VALUE constants representing the range of values for this type.
- All integer literals in Java are 32−bit int values unless the value is followed by l or L as in 235L, indicating the value should be interpreted as a long.

Related Topics

byte Java Keyword

int Java Keyword

short Java Keyword

native Java Keyword

The native keyword may be applied to a method to indicate that the method is implemented in a language other then Java.

Examples

```
  native String getProcessorType();
```

Remarks

- Native methods are beyond the scope of this documentation.

Related Topics

None.

new Java Keyword

The new keyword is used to create a new instance of a class.

Examples

```
  String sName = new String();
  Float fVal = new Float(0.15);
```

Remarks

- The argument following the new keyword must be a class name followed by a series of constructor arguments in required parentheses.
- The collection of arguments must match the signature of a constructor for the class.
- The type of the variable on the left side of the = must be assignment−compatible with the class or interface being instantiated.

Related Topics

None.

null Java Keyword

null is a Java reserved word representing no value.

Examples

```
Integer i;
i = null;


String s;
if (s != null)
{
   <statements>
}
```

Remarks

- Assigning null to a non−primitive variable has the effect of releasing the object to which the variable previously referred.
- null cannot be assigned to variables of primitive types (byte, short, int, long, char, float, double, boolean)

Related Topics

None.

package Java Keyword

The package keyword specifies the Java package in which the classes declared in a Java source file reside.

Examples

```
package com.mycompany;

public class MyClass
{
}
```

Remarks

- The package statement, if present, must be the first non−comment text in a Java source file.
- In the example above, the fully−qualified class name of the MyClass class is com.mycompany.MyClass.
- If a Java source file does not contain a package statement, the classes defined in the file are in the *default package*. Note that classes in the default package may not be referenced from classes in non−default packages.

Related Topics

import Java Keyword

private Java Keyword

The private keyword is an *access control modifier* that may be applied to a class, a method or a field (a variable declared in a class).

Examples

```
public class MyPublicClass
{
   private class MyPrivateClass
   {
   }

   private int i;

   private String myMethod()
   {
     <statements>
   }
}
```

Remarks

- A private (inner) class, method or field may only be referenced from within the class in which it is declared. It is not visible outside the class or to subclasses.
- The default access for all class members is *package access*, meaning that unless a specific access control modifier is present the class members are accessible from within any class in the same package.

Related Topics

protected Java Keyword

public Java Keyword

protected Java Keyword

The protected keyword is an *access control modifier* that may be applied to a class, a method or a field (a variable declared in a class).

Examples

```
public class MyPublicClass
{
   protected class MyPrivateClass
   {
   }

   protected int i;

   protected String myMethod()
   {
     <statements>
   }
}
```

Remarks

- A protected class, method or field may be referenced from within the class in which it is declared, any other classes in the same package, and any subclasses regardless of the package in which a subclass is declared.
- The default access for all class members is *package access*, meaning that unless a specific access control modifier is present the class members are accessible from within any class in the same package.

Related Topics

private Java Keyword

public Java Keyword

public Java Keyword

The public keyword is an *access control modifier* that may be applied to a class, a method or a field (a variable declared in a class).

Examples

```
public class MyPublicClass
{
   public class MyPrivateClass
   {
   }

   public int i;

   public String myMethod()
   {
     <statements>
   }
}
```

Remarks

- A public class, method or field may only be referenced from any other class or package.
- The default access for all class members is *package access*, meaning that unless a specific access control modifier is present the class members are accessible from within any class in the same package.

Related Topics

private Java Keyword

protected Java Keyword

return Java Keyword

The return keyword causes a method to return to the method that called it, passing a value that matches the return type of the returning method.

Examples

```
public void myVoidMethod()
{
   <statements>
   return;
}


public String myStringMethod()
{
   String s = "my response";
   return s;
}


public int myIntMethod()
{
   int i = 5;
   return(i);
}
```

Remarks

- If the method has a non−void return type, the return statement must have an argument of the same or a compatible type.
- The parentheses surrounding the return value are optional.

Related Topics

None.

short Java Keyword

short is a Java primitive type.

A short variable may store a 16−bit signed integer.

Examples

```
short number = 5;
short octalNumber = 0077;
short hexNumber = 0xff;
```

Remarks

- The Short class is a wrapper class for the short primitive type. It defines MIN_VALUE and MAX_VALUE constants representing the range of values for this type.
- All integer literals in Java are 32−bit int values unless the value is followed by l or L as in 235L, indicating the value should be interpreted as a long.

Related Topics

byte Java Keyword

int Java Keyword

long Java Keyword

static Java Keyword

The static keyword may be applied to an inner class (a class defined within another class), method or field (a member variable of a class).

Examples

```
public class MyPublicClass
{
   public final static int MAX_OBJECTS = 100;
   static int _numObjects = 0;

   static class MyStaticClass
   {
   }

   static int getNumObjects()
   {
   }
}
```

Remarks

- In general, the static keyword means that the entity to which it is applied is available outside any particular instance of the class in which the entity is declared.
- A static (inner) class may be instantiated and reference by other classes as though it were a top−level class. In the example above, code in another class could instantiate the MyStaticClass class by qualifiying it's name with the containing class name, as MyClass.MyStaticClass.
- A static field (member variable of a class) exists once across all instances of the class.
- A static method may be called from outside the class without first instantiating the class. Such a reference always includes the class name as a qualifier of the method call. In the example above code outside the MyClass class would invoke the getNumObjects() static method as MyClass.getNumObjects().
- The pattern:

  public final static <type> varName = <value>;

  is commonly used to declare class constants that may be used from outside the class. A reference to such a constant is qualified with the class name. In the example above, another class could reference the MAX_OBJECTS constant as MyClass.MAX_OBJECTS.

Related Topics

final Java Keyword

super Java Keyword

The super keyword refers to the superclass of the class in which the keyword is used.

Examples

```
public class MyClass
{
   public MyClass(String arg)
   {
      super(arg);
   }

   public String myStringMethod()
   {
      return super.otherStringMethod();
   }
```

Remarks

- super as a standalone statement represents a call to a constructor of the superclass.
- super.<methodName>() represents a call to a method of the superclass. This usage is only necessary when calling a method that is overridden in this class in order to specify that the method should be called on the superclass.

Related Topics

None.

switch Java Keyword

The switch statement is used to select execution of one of multiple code blocks based on an expression.

Examples

```
int arg = <some value>;
switch (arg)
{
   case 1:
      <statements>
      break;
   case 2:
      <statements>
      break;
   default:
      <statements>
      break;
}


char arg = <some value>;
switch (arg)
{
   case 'y':
   case 'Y':
      <statements>
      break;
   case 'n':
   case 'N':
      <statements>
      break;
   default:
      <statements>
      break;
}
```

Remarks

- The switch condition must evaluate to a byte, char, short or int.
- A case block does not have an implicit ending point. A break statement is typically used at the end of each case block to exit the switch statement.
- Without a break statement, the flow of execution will flow into all following case and/or default blocks.

Related Topics

break Java Keyword

case Java Keyword

default Java Keyword

synchronized Java Keyword

The synchronized keyword may be applied to a method or statement block and provides protection for *critical sections* that should only be executed by one thread at a time.

Examples

```
public class MyClass
{
   public synchronized static String mySyncStaticMethod()
   {
   }

   public synchronized String mySyncMethod()
   {
   }
{


public class MyOtherClass
{
   Object someObj;

   public String myMethod()
   {
      <statements>
      synchronized (someObj)
      {
         <statements affecting someObj>
      }
   }
}
```

Remarks

- The synchronized keyword prevents a critical section of code from being executed by more than one thread at a time.
- When applied to a static method, as with MySyncStaticMethod in the examples above, the entire class is locked while the method is being executed by one thread at a time.
- When applied to an instance method, as with MySyncMethod in the examples above, the instance is locked while being accessed by one thread at at time.
- When applied to an object or array, the object or array is locked while the associated code block is executed by one thread at at time.

Related Topics

None.

this Java Keyword

The this keyword refers to the current instance.

Examples

```
public class MyClass
{
   int number;

   public MyClass(int number)
   {
      this.number = number;
   }
}
```

Remarks

- The this keyword is used to refer to the current instance when a reference may be ambiguous.
- In the example above, the constructor argument number has the same name as a member variable of the class. this.number means specifically the number member variable of this instance of MyClass.

Related Topics

None.

throw Java Keyword

The throw keyword is used to raise an exception.

Examples

```
import java.io.IOException;

public class MyClass
{
   public method readFile(String filename) throws IOException
   {
      <statements>
      if (error)
      {
         throw new IOException("error reading file");
      }
   }
}
```

Remarks

- The throw statement takes a java.lang.Throwable as an argument. The Throwable is propagated up the call stack until it is caught by an appropriate catch block.
- Any method that throws an exception that is not a RuntimeException must also declare the exceptions it throws using a throws modifier on the method declaration.

Related Topics

catch Java Keyword

finally Java Keyword

throws Java Keyword

try Java Keyword

throws Java Keyword

The throws keyword may be applied to a method to indicate the method raises particular types of exceptions.

Examples

```
import java.io.IOException;

public class MyClass
{
   public method readFile(String filename) throws IOException
   {
      <statements>
      if (error)
      {
         throw new IOException("error reading file");
      }
   }
}
```

Remarks

- The throws keyword takes a comma−separated list of java.lang.Throwables as an argument.
- Any method that throws an exception that is not a RuntimeException must also declare the exceptions it throws using a throws modifier on the method declaration.
- The caller of a method with a throws clause is required to enclose the method call in a try−catch block.

Related Topics

catch Java Keyword

finally Java Keyword

throws Java Keyword

try Java Keyword

transient Java Keyword

The transient keyword may be applied to member variables of a class to indicate that the member variable should not be serialized when the containing class instance is serialized.

Examples

```
public class MyClass
{
   private transient String password;
}
```

Remarks

Related Topics

None.

try Java Keyword

The try keyword is used to enclose blocks of statements that might throw exceptions.

Examples

```
try
{
   <block that may throw exceptions>
}
catch (<java.lang.Exception or subclass> e)
{
   <code to handle exception e>
}


try
{
   <block that may throw different exceptions>
}
catch (FooException e)
{
   <code to handle FooException e>
}
catch (BarException e)
{
   <code to handle BarException e>
}


try
{
   <block that may throw exceptions>
}
catch (<java.lang.Exception or subclass> e)
{
   <code to handle exception e>
}
finally
{
   <statements that execute with or without exception>
}
```

Remarks

- Every try block must have at least one catch or finally clause.
- If a particular exception class is not handled by any catch clause, the exception propagates up the call stack to the next enclosing try block, recursively. If an exception is not caught by any enclosing try block, the Java interpretor will exit with an error message and stack trace.

Related Topics

catch Java Keyword

finally Java Keyword

true Java Keyword

The true keyword represents one of the two legal values for a boolean variable.

Examples

```
  boolean isComplete = true;
```

Remarks

None.

Related Topics

boolean Java Keyword

false Java Keyword

void Java Keyword

The void keyword represents a null type.

Examples

```
public class MyClass
{
   public void doSomething();
   {
      <statements>
      return;
   }
}
```

Remarks

- void may be used as the return type of a method to indicate the method does not return a value.

Related Topics

None.

volatile Java Keyword

The volatile keyword may be used to indicate a member variable that may be modified asynchronously by more than one thread.

Note: the volatile keyword is not implemented in many Java Virtual Machines.

Examples

```
public class MyClass
{
   volatile int sharedValue;
}
```

Remarks

- volatile is intended to guarantee that all threads see the same value of the specified variable.

Related Topics

None.

while Java Keyword

The while keyword specifies a loop that is repeated as long as a condition is true.

Examples

```
while (!found)
{
   <statements>
}
```

Remarks

None.

Related Topics

break Java Keyword

continue Java Keyword

do Java Keyword

for Java Keyword