

• OBJECTS AND CLASSES

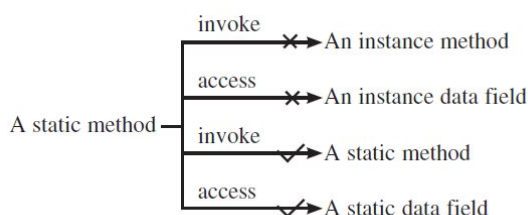
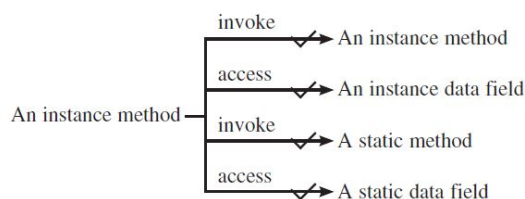
- 文件中只能有一个类是public，且必须与文件同名。
- A class may be defined without constructors. In this case, a public no-arg constructor with an empty body is implicitly defined in the class. This constructor, called a default constructor, is provided automatically only if no constructors are explicitly defined in the class.
 - 默认构造方法，当且仅当类中没有明确定义任何构造方法时才会自动提供它。
- 如果一个引用类型的数据域没有引用任何对象，那么这个数据域就有一个特殊的Java值null

```
class Student {  
    String name; // name has the default value null  
    int age; // age has the default value 0  
    boolean isScienceMajor; // isScienceMajor has default value false  
    char gender; // gender has default value '\u0000'  
}
```

- 但是，Java没有给方法中的局部变量赋默认值，x&y没有被初始化，编译错误

```
class Test {  
    public static void main(String[] args) {  
        int x; // x has no default value  
        String y; // y has no default value  
        System.out.println("x is " + x);  
        System.out.println("y is " + y);  
    }  
}
```

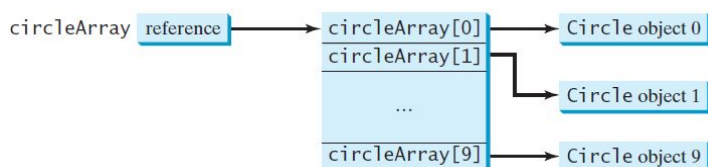
- For a variable of a primitive type, the value is of the primitive type. For a variable of a reference type, the value is a reference to where an object is located.
 - 基本类型->对应内存的值就是它的值
 - 引用类型->对应内存的值是对象的地址
- A static method cannot access instance members of the class.
- 实例->实例&静态，静态->静态



• 可见性修饰符

作用域	当前类	同一package	子孙类	其他package
public	√	√	√	√
protected	√	√	√	×
default (无修饰词)	√	√	×	×
private	√	×	×	×

- Passing an object to a method is to pass the reference of the object.
- Java uses exactly one mode of passing arguments: pass-by-value.
- Pass-by-value on references can be best described semantically as pass-by-sharing; that is, the object referenced in the method is the same as the object being passed.
- An array of objects is actually an array of reference variables引用变量的数组. So, invoking `circleArray[1].getArea()` involves two levels of referencing, as shown in Figure 9.19. `circleArray` references the entire array; `circleArray[1]` references a `Circle` object. 引用整个数组；引用一个对象



- Immutable Objects and Classes不可变对象、不可变类
 - 没有一个返回指向可变数据域的引用的访问器方法

For a class to be immutable, it must meet the following requirements:

- All data fields must be private.
- There can't be any mutator methods for data fields.
- No accessor methods can return a reference to a data field that is mutable.

- The scope of instance and static variables is the entire class, regardless of where the variables are declared.
 - 实例变量和静态变量的作用域是整个类，无论变量是在哪里声明的
 - Instance and static variables in a class are referred to as the class' s variables or data fields. A variable defined inside a method is referred to as a local variable.
- 构造方法中，`this()`应在其他语句之前



Note

Java requires that the `this(arg-list)` statement appear first in the constructor before any other executable statements.

• OBJECT-ORIENTED THINKING

- The instances of all wrapper classes are immutable; this means that, once the objects are created, their internal values cannot be changed. 所有包装类的实例是不

可改变的

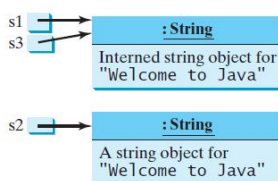
- A primitive type value can be automatically converted to an object using a wrapper class, and vice versa, depending on the context. 基本数据类型可使用包装类自动转换成一个对象，反过来也可以

- Converting a primitive value to a wrapper object is called boxing.
- The reverse conversion is called unboxing.

```
Test.java:7: 错误: 不兼容的类型: int无法转换为Double
    Double k = 3;
```

- An instance of BigInteger can represent an integer of any size. You can use new BigInteger(String) and new BigDecimal(String) to create an instance of BigInteger and BigDecimal. 它们都是不可变的。
- String
 - A String object is immutable
 - the JVM uses a unique instance for string literals with the same character sequence. Such an instance is called interned.

```
String s1 = "Welcome to Java";
String s2 = new String("Welcome to Java");
String s3 = "Welcome to Java";
System.out.println("s1 == s2 is " + (s1 == s2));
System.out.println("s1 == s3 is " + (s1 == s3));
```




display

```
s1 == s2 is false
s1 == s3 is true
```

- A new object is created if you use the new operator.
- If you use the string initializer, no new object is created if the interned object is already created.
- 对于String s = "java"，在编译成.class时能够识别为同一字符串的,自动优化成常量,所以如果有多个字符串"java"，则它们都会引用自同一String对象。
- 常量池
 - 利用缓存机制实现常量池：为了减少不必要的内存消耗和内存开辟次数，Integer 里做了一个缓存，缓存了从-128 到127 之间的Integer 对象，总共是256个对象。
- T F T T

```
Integer i11 = 126;
Integer i12 = 126;
Integer i13 = new Integer(126);
Integer i14 = 125+1;
Integer i15 = 125 + new Integer(1);
```



```
System.out.println(i11==i12);
System.out.println(i11==i13);
System.out.println(i11==i14);
System.out.println(i11==i15);
```

```
Integer i11 = Integer.valueOf(126);
Integer i12 = Integer.valueOf(126);
Integer i13 = new Integer(126);
Integer i14 = Integer.valueOf(126);
Integer i15 = Integer.valueOf(125 + (new Integer(1)).intValue());
```

- 尽量使用Integer.valueOf(int) 方法来获取一个Integer 对象，而不使用new Integer(int) 来构造Integer 对象。

```
Integer a = new Integer(3);
Integer b = 3; ← valueOf(3)
int c = 3;
System.out.println(a == b);
System.out.println(a == c);
```

intValue(a)

false
true

- StringBuilder&StringBuffer
 - the buffer in StringBuffer are synchronized.
 - 多任务并发访问：StringBuffer。
 - The new array size is 2 * (the previous arraysize + 1).
 - ATTENTION!!

```
String s = "Java";
StringBuilder builder = new StringBuilder(s);
change(s, builder);
System.out.println(s);
System.out.println(builder);

private static void change(String s, StringBuilder builder) {
    s = s + " and HTML";
    builder.append(" and HTML");
}
```

• INHERITANCE AND POLYMORPHISM

- The keyword super refers to the superclass and can be used to invoke the superclass's methods and constructors.
- The statement super() or super(arguments) must be the first statement of the subclass' s constructor; this is the only way to explicitly invoke a superclass constructor.
 - 出现在子类构造方法的第一行
- A constructor may invoke an overloaded constructor or its superclass constructor. If neither is invoked explicitly, the compiler automatically puts super() as the first statement in the constructor.

<pre>public ClassName() { // some statements }</pre>	Equivalent	<pre>public ClassName() { super(); // some statements }</pre>
--	------------	---

- If possible, you should provide a no-arg constructor for every class to make the class easy to extend and to avoid errors.
- Overriding vs. Overloading

- <https://www.geeksforgeeks.org/can-we-overload-or-override-static-methods-in-java/>

- Can we overload static methods?
 - The answer is 'Yes' . We can have two or more static methods with same name, but differences in input parameters.
- Can we overload methods that differ only by static keyword?
 - NO
- Can we Override static methods in java?
 - NO

- Overloading means to define multiple methods with the same name but different signatures. Overriding means to provide a new implementation for a method in the subclass.

- Overriding

- To override a method, the method must be defined in the subclass using the same signature and the same return type as in its superclass.

- 方法重写 (method overriding) : 子类 and 父类一样的签名和一样的返回值类型
- 如果子类定义的方法在父类是私有的, 则这两个方法完全没有关系
- Like an instance method, a static method can be inherited. **However, a static method cannot be overridden.** If a static method defined in the superclass is redefined in a subclass, *the method defined in the superclass is hidden*. The hidden static methods can be invoked using the syntax `SuperClassName.staticMethodName`.

- OverLoading

- Overloaded methods must have different parameter lists. You cannot overload methods based on different modifiers or return types.
 - 不能基于不同修饰符或返回值类型来重载。
- Sometimes there are two or more possible matches for the invocation of a method, but the compiler cannot determine the best match. This is referred to as ambiguous invocation. Ambiguous invocation causes a compile error.多个匹配时, 歧义调用, 编译错误
 - Both `max(int, double)` and `max(double, int)` are possible candidates to match `max(1, 2)`. Because neither is better than the other, the invocation is ambiguous, resulting in a compile error.

- 多态

- 多态意味着父类变量可以指向子类对象
- 动态绑定 (对于重写)
 - declared type and actual type

```
Object o = new GeometricObject();
System.out.println(o.toString());
```

- Here `o`'s declared type is `Object`.

- Here o' s actual type is GeometricObject
- 从特殊到一般查找
- 对象转换
 - java.lang.ClassCastException: Fruit cannot be cast to Apple

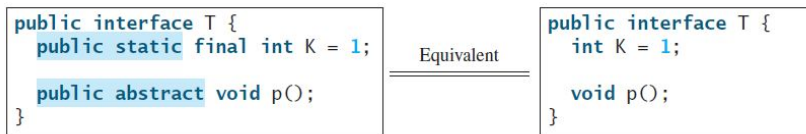
```
public class Test {
    public static void main(String[] args) {
        Object fruit = new Fruit();
        Object apple = (Apple)fruit;
    }
}
class Apple extends Fruit {
}
class Fruit {
```

- The output is **false** if the Circle class in (a) is used. The Circle class has two *overloaded* methods: equals(Circle circle) defined in the Circle class and equals(Object circle) defined in the Object class, inherited by the Circle class. At compilation time, circle1.equals(circle2) is matched to equals(Object circle), because the declared type for circle1 and circle2 is Object.

```
public class Test {
    public static void main(String[] args) {
        Object circle1 = new Circle();
        Object circle2 = new Circle();
        System.out.println(circle1.equals(circle2));
    }
}
class Circle {
    double radius;
    public boolean equals(Circle circle) {
        return this.radius == circle.radius;
    }
}
```

- 存储在ArrayList中的元素必须是一种对象。
- A subclass may override a protected method defined in its superclass and change its visibility to public. However, a subclass cannot weaken the accessibility of a method defined in the superclass. For example, if a method is defined as public in the superclass, it must be defined as public in the subclass. 子类可以重写父类的protected方法，并把可见性改为public。
- final类不能被继承，final方法不能被重写。
- Abstract Classes and Interfaces
 - 抽象类
 - 抽象方法不能包含在非抽象类中。如果抽象父类的子类不能实现所有的抽象方法，那么子类必须是抽象的。
 - 抽象方法是非静态的。
 - 即使子类的父类是具体的，子类也可以是抽象的。
 - 抽象类可以用作一种数据类型。
 - 接口
 - 只包含常量和抽象方法

- Since all data fields are public static final and all methods are public abstract in an interface, Java allows these modifiers to be omitted. Therefore the following interface definitions are equivalent



- 所有数据域 : public static final
- 所有方法 : public abstract
- All methods defined in an interface are public. When a class implements the interface, the method must be declared public. The visibility cannot be reduced.

i Show the error in the following code:

```
interface A {
    void m1();
}

class B implements A {
    void m1() {
        System.out.println("m1");
    }
}
```

• GENERICS

- Generics enable you to detect errors at compile time rather than at runtime. 在编译时而不是运行时检测出错误
- Generic types must be reference types. You cannot replace a generic type with a primitive type such as int, double, or char.
- Generic Methods 泛型方法
 - A generic type can be defined for a static method. 可以为静态方法定义泛型类型
 - To declare a generic method, you place the generic type <E> immediately after the keyword static in the method header. For example,

```
public static <E> void print(E[] list)
```

To invoke a generic method, prefix the method name with the actual type in angle brackets. i
For example,

```
GenericMethodDemo.<Integer>print(integers);
GenericMethodDemo.<String>print(strings);
```

or simply invoke it as follows:

```
print(integers);
print(strings);
```

- A generic type can be specified as a subtype of another type. Such a generic type is called bounded.

3 <E extends GeometricObject>

- Wildcard Generic Types 通配泛型

- You can use unbounded wildcards (非受限通配), bounded wildcards (受限通配), or lower-bound wildcards (下限通配) to specify a range for a generic type.
- The fact is that Integer is a subtype of Number, but GenericStack<Integer> is not a subtype of GenericStack<Number>

```
public class WildCardNeedDemo {
    public static void main(String[] args) {
        GenericStack<Integer> intStack = new GenericStack<>();
        intStack.push(1); // 1 is autoboxed into new Integer(1)
        intStack.push(2);
        intStack.push(-2);

        System.out.print("The max number is " + max(intStack));
    }

    /** Find the maximum in a stack of numbers */
    public static double max(GenericStack<Number> stack) {
        double max = stack.pop().doubleValue(); // Initialize max

        while (!stack.isEmpty()) {
            double value = stack.pop().doubleValue();
            if (value > max)
                max = value;
        }

        return max;
    }
}
```

- The first form, ?, called an unbounded wildcard, is the same as ? extends Object.
- The second form, ? extends T, called a bounded wildcard, represents T or a subtype of T.
- The third form, ? super T, called a lower-bound wildcard, denotes T or a supertype of T.
- <? extends Number> is a wildcard type that represents Number or a subtype of Number,

You can fix the error by replacing line 12 in Listing 19.7 as follows:

```
public static double max(GenericStack<? extends Number> stack) {
```

- The inheritance relationship involving generic types and wildcard types is summarized in Figure 19.6. In this figure, A and B represent classes or interfaces, and E is a generic type parameter.

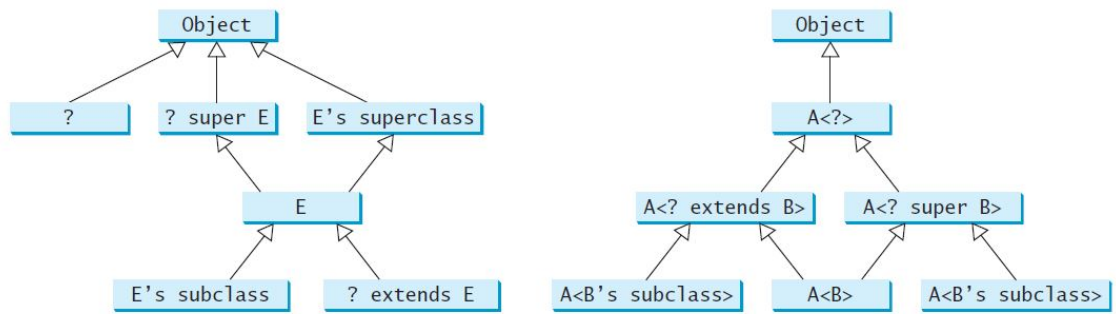


FIGURE 19.6 The relationship between generic types and wildcard types.

- Erasure and Restrictions on Generics

- 不管实际的具体 类型是什么，泛型类是被它所有实例所共享的。

```
ArrayList<String> list1 = new ArrayList<>();
ArrayList<Integer> list2 = new ArrayList<>();
```

- Although `ArrayList<String>` and `ArrayList<Integer>` are two types at compile time, only one `ArrayList` class is loaded into the JVM at runtime. `list1` and `list2` are both instances of `ArrayList`, so the following statements display true:

```
System.out.println(list1 instanceof ArrayList);
System.out.println(list2 instanceof ArrayList);
```

- However, the expression `list1 instanceof ArrayList<String>` is wrong. Since `ArrayList<String>` is not stored as a separate class in the JVM, using it at runtime makes no sense.
- **Restriction 1: Cannot Use `new E()`**
- **Restriction 2: Cannot Use `new E[]`**
 - You cannot create an array using a generic type parameter. For example, the following statement is wrong:

```
E[] elements = new E[capacity];
```

- You can circumvent this limitation by creating an array of the `Object` type and then casting it to `E[]`, as follows:

```
E[] elements = (E[])new Object[capacity];
```

- Generic array creation using a generic class is not allowed, either. For example, the following code is wrong:

```
ArrayList<String>[] list = new ArrayList<String>[10];
```

- You can use the following code to circumvent this restriction: However, you will still get a compile warning.

```
ArrayList<String>[] list = (ArrayList<String>[])new  
ArrayList[10];
```

- **Restriction 3: A Generic Type Parameter of a Class Is Not Allowed in a Static Context**

- Since all instances of a generic class have the same runtime class, the static variables and methods of a generic class are shared by all its instances. Therefore, it is illegal to refer to a generic type parameter for a class in a static method, field, or initializer. (泛型类)

由于泛型类的所有实例都有相同的运行时类，所以泛型类的静态变量和方法是被它的所有实例所共享的。因此，在静态方法、数据域或者初始化语句中，为类引用泛型类型参数是非法的。例如，下面的代码是非法的：

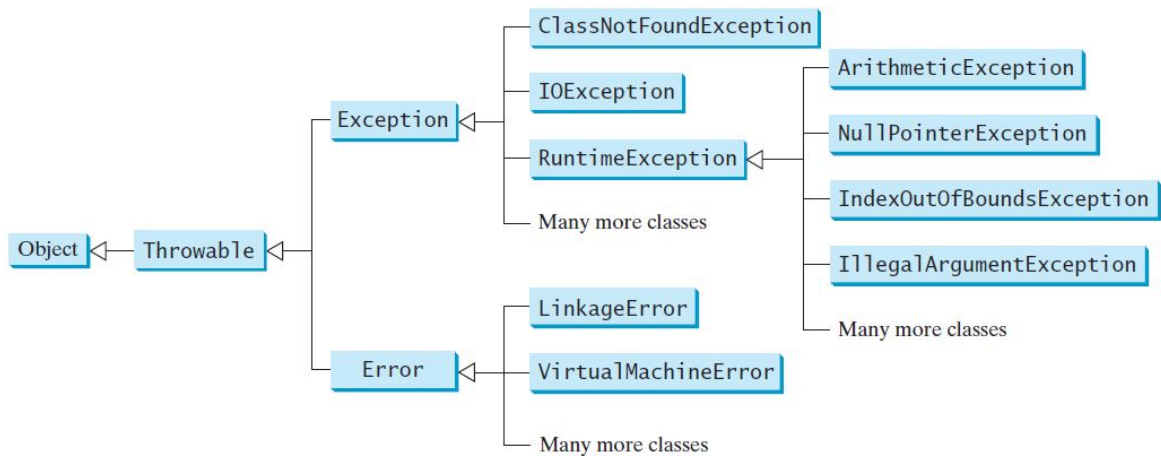
```
public class Test<E> {  
    public static void m(E o1) { // Illegal  
    }  
  
    public static E o1; // Illegal  
  
    static {  
        E o2; // Illegal  
    }  
}
```

- Can a method that uses a generic class parameter be static? Why?使用泛型类作为参数的方法可以是静态的吗
 - NO

- **Restriction 4: Exception Classes Cannot Be Generic**

- **EXCEPTION HANDLING**

- In Java, runtime errors are thrown as exceptions. An exception is an object that represents an error or a condition that prevents execution from proceeding normally. If the exception is not handled, the program will terminate abnormally.如果异常没有被处理，那么程序将会非正常终止
- Exceptions are thrown from a method. The caller of the method can catch and handle the exception.
- Exception Types异常类型
 - The root class for exceptions is java.lang.Throwable.
 - The class names Error, Exception, and RuntimeException are somewhat confusing. All three of these classes are exceptions, and all of the errors occur at runtime.



- The exception classes can be classified into three major types: system errors, exceptions, and runtime exceptions :
 - system errors
 - System errors are thrown by the JVM and are represented in the Error class. The Error class describes internal system errors, though such errors rarely occur.
 - exceptions
 - Exceptions are represented in the Exception class, which describes errors caused by your program and by external circumstances.
 - runtime exceptions
 - describes programming errors
- RuntimeException, Error, and their subclasses are known as unchecked exceptions. All other exceptions are known as checked exceptions,
- Unchecked exceptions can occur anywhere in a program.
- A handler for an exception is found by propagating the exception backward through a chain of method calls, starting from the current method.
 - If a method does not declare exceptions in the superclass, you cannot override it to declare exceptions in the subclass. 如果方法没有在父类中声明异常，那么就不能在子类中对其进行继承来声明异常。
- 捕获异常：
 - If no exceptions arise during the execution of the try block, the catch blocks are skipped.
 - If one of the statements inside the try block throws an exception, Java skips the remaining statements in the try block
 - Each catch block is examined in turn, from first to last, to see whether the type of the exception object is an instance of the exception class in the catch block.
 - If no handler is found, Java exits this method, passes the exception to the method that invoked the method, and continues the same process to find a handler
 - If no handler is found in the chain of methods being invoked, the program terminates and prints an error message on the console.

- The order in which exceptions are specified in catch blocks is important. A compile error will result if a catch block for a superclass type appears before a catch block for a subclass type. 父类必须在子类之后
- Can you throw multiple exceptions in one throw statement?
 - NO
- Does the presence of a try-catch block impose overhead when no exception occurs?
 - NO
- The finally Clause (finally子句)
 - The finally clause is always executed regardless whether an exception occurred or not.
 - The finally block executes even if there is a return statement prior to reaching the finally block.
 - The catch block may be omitted when the finally clause is used. 使用finally子句时可以省略catch块
- Rethrowing Exceptions
 - The statement throw ex rethrows the exception to the caller

```
try {
    statements;
}
catch (TheException ex) {
    perform operations before exits;
    throw ex;
}
```

- Suppose that statement2 causes an exception in the following statement:

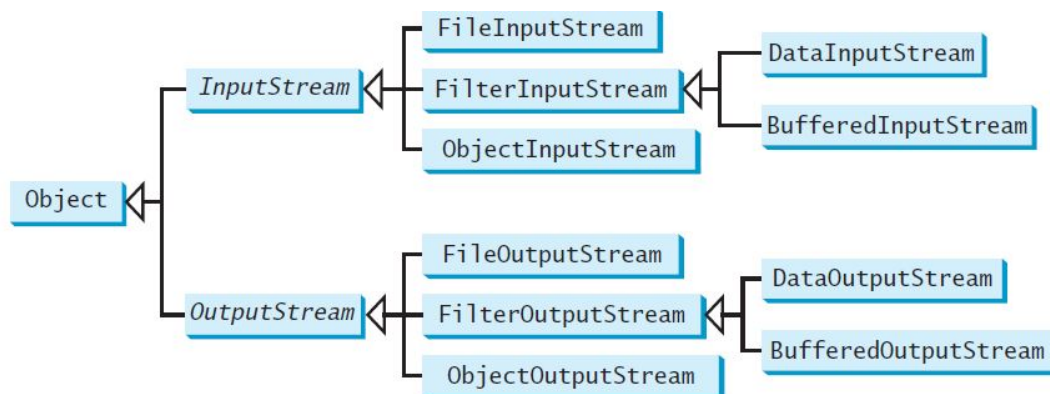
```
try {
    statement1;
    statement2;
    statement3;
}
catch (Exception1 ex1) {
}
catch (Exception2 ex2) {
    throw ex2;
}
finally {
    statement4;
}
statement5;
```

- ■ If no exception occurs, will statement4 be executed, and will statement5 be executed?
 - YES YES
- ■ If the exception is of type Exception1, will statement4 be executed, and will statement5 be executed?
 - YES YES
- ■ If the exception is of type Exception2, will statement4 be executed, and will statement5 be executed?

- YES NO
- ■ If the exception is not Exception1 nor Exception2, will statement4 be executed, and will statement5 be executed?
- YES NO

• BINARY I/O

- input -> 读文件 output -> 写文件
- The abstract InputStream is the root class for reading binary data, and the abstract OutputStream is the root class for writing binary data.

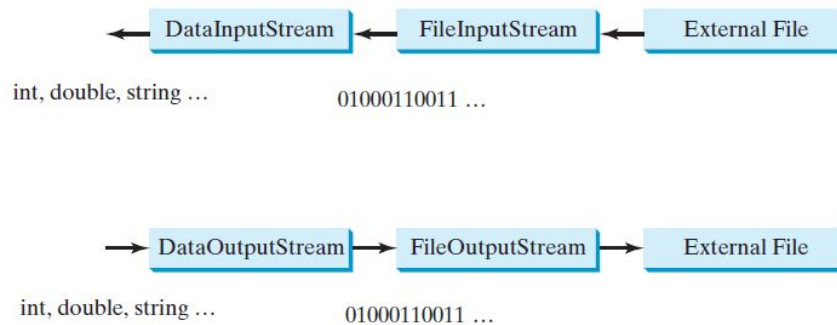


- All the methods in the binary I/O classes are declared to throw `java.io.IOException` or a subclass of `java.io.IOException`.
- `FileInputStream/FileOutputStream`
 - 没有引入新方法
 - 如果试图为一个不存在的文件创建 `FileInputStream` 对象，将会发生 `java.io.FileNotFoundException` 异常。
- `FilterInputStream/FilterOutputStream`
 - 使用过滤器类就可以读取整数值、双精度值和字符串，而不是字节或字符
- `DataInputStream/DataOutputStream`
 - 实现了 `DataInput/DataOutput` 接口
- Characters and Strings in Binary I/O
 - A Unicode character consists of two bytes.
 - The `writeChar(char c)` method writes the Unicode of character `c` to the output.
 - The `writeChars(String s)` method writes the Unicode for each character in the string `s` to the output.
 - The `writeBytes(String s)` method writes the lower byte of the Unicode for each character in the string `s` to the output. The high byte of the Unicode is discarded. 低字节写入，高字节丢弃，适合ASCII
 - The `writeUTF(String s)` method writes two bytes of length information to the output stream, followed by the modified UTF-8 representation of every character in the string `s`. UTF-8具有存储每个ASCII码就节省一个字节的优势。

- Creating DataInputStream/DataOutputStream
- DataInputStream/DataOutputStream are created using the following constructors

```
public DataInputStream(InputStream instream)
public DataOutputStream(OutputStream outstream)
```

- You can view DataInputStream/FileInputStream and DataOutputStream/FileOutputStream working in a pipe line



- Detecting the End of a File
 - If you keep reading data at the end of an InputStream, an EOFException will occur
- BufferedInputStream/BufferedOutputStream
 - 没有包含新的方法
 - 缓冲区默认512B
 - 提高效率：

```
DataOutputStream output = new DataOutputStream(
    new BufferedOutputStream(new FileOutputStream("temp.dat")));
DataInputStream input = new DataInputStream(
    new BufferedInputStream(new FileInputStream("temp.dat")));
```

- What is wrong in the following code?

```
import java.io.*;
public class Test {
    public static void main(String[] args) {
        try {
            FileInputStream fis = new FileInputStream("test.dat"); } {
        }
        catch (IOException ex) {
            ex.printStackTrace();
        }
        catch (FileNotFoundException ex) {
            ex.printStackTrace();
        }
    }
}
```

- the catch clause for java.io.FileNotFoundException should be put before the catch clause for java.io.IOException.
- Suppose you run the following program on Windows using the default ASCII encoding after the program is finished, how many bytes are there in the file

t.txt?

```
public class Test {
    public static void main(String[] args)
        throws java.io.IOException {
        try (java.io.PrintWriter output =
            new java.io.PrintWriter("t.txt"); ) {
            output.printf("%s", "1234");
            output.printf("%s", "5678");
            output.close();
        }
    }
}
```

• 8

- After the following program is finished, how many bytes are there in the file t.dat?

```
import java.io.*;

public class Test {
    public static void main(String[] args) throws IOException {
        try (DataOutputStream output = new DataOutputStream(
            new FileOutputStream("t.dat")); ) {
            output.writeInt(1234);
            output.writeInt(5678);
            output.close();
        }
    }
}
```

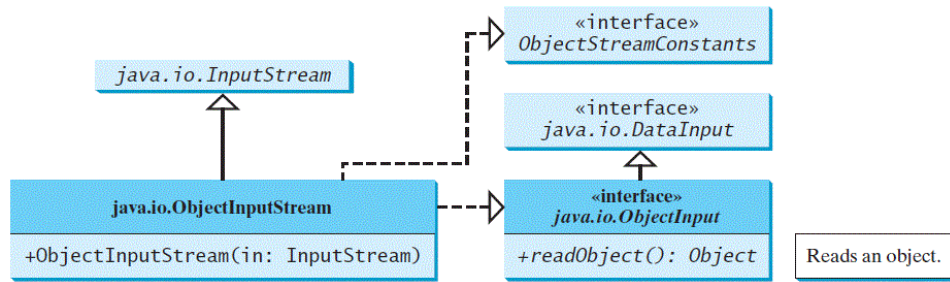
• 8

- For each of the following statements on a DataOutputStream output, how many bytes are sent to the output?

```
output.writeChar('A');
output.writeChars("BC");
output.writeUTF("DEF");
```

• 2 4 2+3

- Object I/O
 - ObjectInputStream/ObjectOutputStream classes can be used to read/write serializable objects.
 - Since ObjectInputStream/ ObjectOutputStream contains all the functions of DataInputStream/ DataOutputStream, you can replace DataInputStream/DataOutputStream completely with ObjectInputStream/ObjectOutputStream. object完全代替data
 - ObjectInputStream extends InputStream and implements ObjectInput and ObjectOutputStreamConstants



- A serializable object is an instance of the `java.io.Serializable` interface, so the object's class must implement `Serializable`.
- The `Serializable` interface is a marker interface. Since it has no methods, you don't need to add additional code in your class that implements `Serializable`.
- If an object is an instance of `Serializable` but contains nonserializable instance data fields, can it be serialized? The answer is no. To enable the object to be serialized, mark these data fields with the `transient` keyword to tell the JVM to ignore them.

```

public class C implements java.io.Serializable {
    private int v1;
    private static double v2;
    private transient A v3 = new A();
}

```

```

class A { } // A is not serializable

```

When an object of the `C` class is serialized, only variable `v1` is serialized. Variable `v2` is not serialized because it is a static variable, and variable `v3` is not serialized because it is marked `transient`. If `v3` were not marked `transient`, a `java.io.NotSerializableException` would occur.