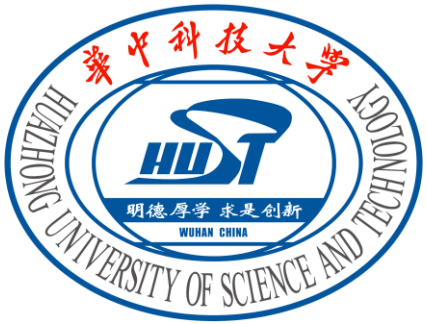
基于Java的面向对象程序设计

陈维亚

weiya_chen@hust.edu.cn

华中科技大学软件学院

第17讲：Java标准库介绍



目录

1. 包与库

❑ Java 包

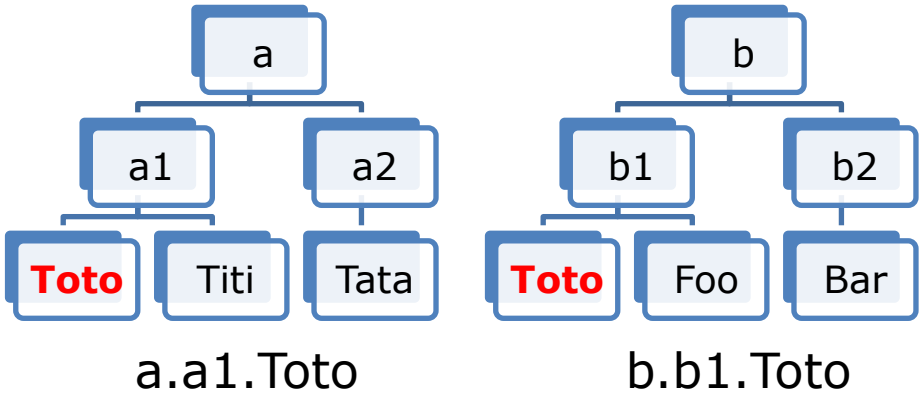
- 避免命名冲突；
- 可以方便查找与使用；
- 访问控制；
- 创建自定义类库，有助于软件重用

分类，多级

❑ 命名

- 包名需全部小写；
- 包名参考文件夹的层级结构；
- 公司一般使用域名反转作为包名

  example.com → com.example.mypackage



1. 包与库

❑ 命名

- 公司一般使用域名反转作为包名

  example.com → com.example.mypackage

制定合法的包名

域名	包名
hyphenated-name.example.org	org.example.hyphenated_name
example.int	int_.example
123name.example.com	com.example._123name

1. 包与库

❑ 使用

```
import java.util.Vector;
```

```
import java.util.*;
```

```
import java.util.Vector;

public class ImportDemo{
    public ImportDemo(){
        Vector newVector = new Vector();

        java.util.ArrayList newList = new java.util.ArrayList();
    }

    public static void main(String args[]){
        new ImportDemo();
    }
}
```

1. 包与库

❑ 使用

```
import java.util.Vector;
```

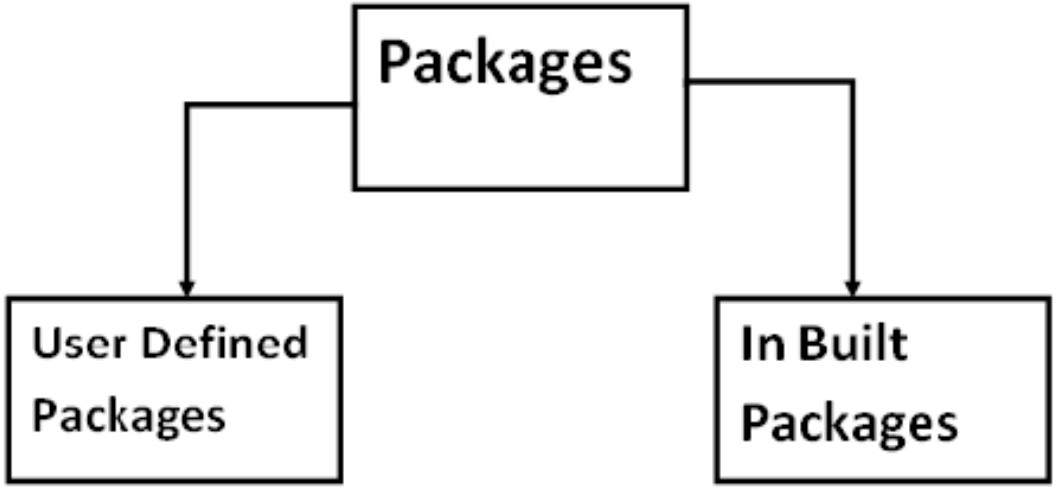
```
import java.util.*;
```

- 包名的声明必须出现在第一行;
- 默认包没有名称;
- 父包与子包：
  - 子包需要显式引用
  - 父包不包含子包

```
import java.awt.*;
import java.awt.color.*;
```

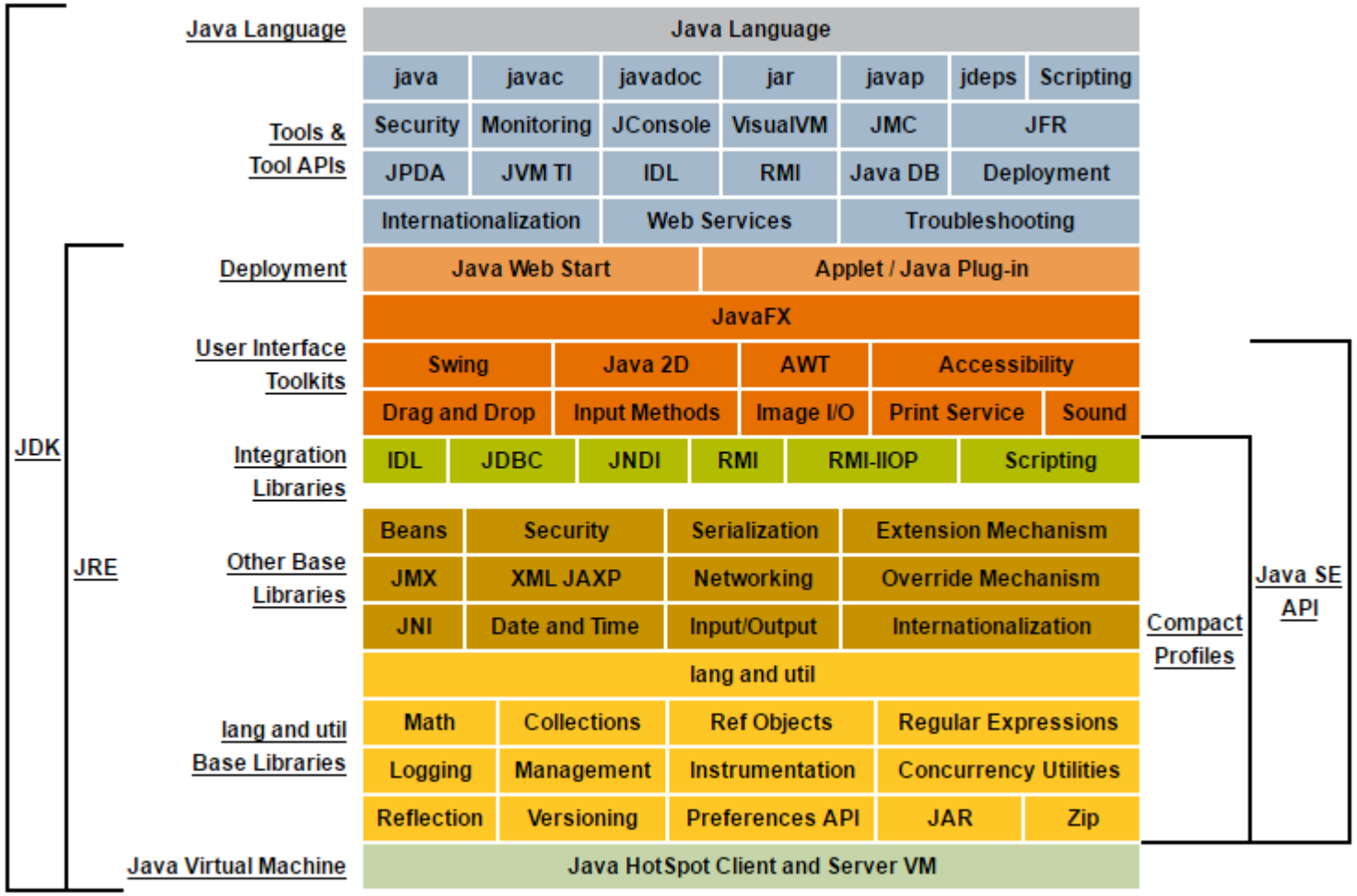
1. 包与库

- 包的种类



1. 包与库

❑ Java API



1. 包与库

❑ Built-in 包

java.lang	String Wrapper classes Math
java.util	Vector Date
java.awt (abstract window toolkit)	Graphics Button Label
java.io	InputStream OutputStream
java.sql	Connection Statement
Javax.swing	JButton JLabel
……	

java.util

- .concurrent
- .jar
- .logging
- .prefs
- .zip

❑ User-defined 包

1. 创建文件夹MyPackage
2. 添加类文件MyClass.java

```
package myPackage;

public class MyClass {
    public void getNames(String s) {
        System.out.println(s);
    }
}
```

```
import myPackage.MyClass;

public class PrintName {
    public static void main(String args[]) {
        String name = "Tom";
        MyClass obj = new MyClass();
        obj.getNames(name);
    }
}
```

1. 包与库

❑ 解决名称冲突

```
import java.util.*;
import java.sql.*;

// we will get a compile time error :
Date today ; //ERROR-- java.util.Date or java.sql.Date?
```

```
import java.util.Date;
import java.sql.*;

java.util.Date deadLine = new java.util.Date();

java.sql.Date today = new java.sql.Date();
```

1. 包与库

❑ CLASSPATH

为了引用 com.zzz.project1.subproject2 包中定义的Circle 类

则Java编译器需要知道 ($BASE_DIR)

"$BASE_DIR\com\zzz\project1\subproject2\Circle.class",

$BASE_DIR 实际上由CLASSPATH环境变量指定

练习1

假如你在默认包中创建了如下几个类：Server、Utilities、Client，现需要把它们放入不同的包中去，如下表所示：

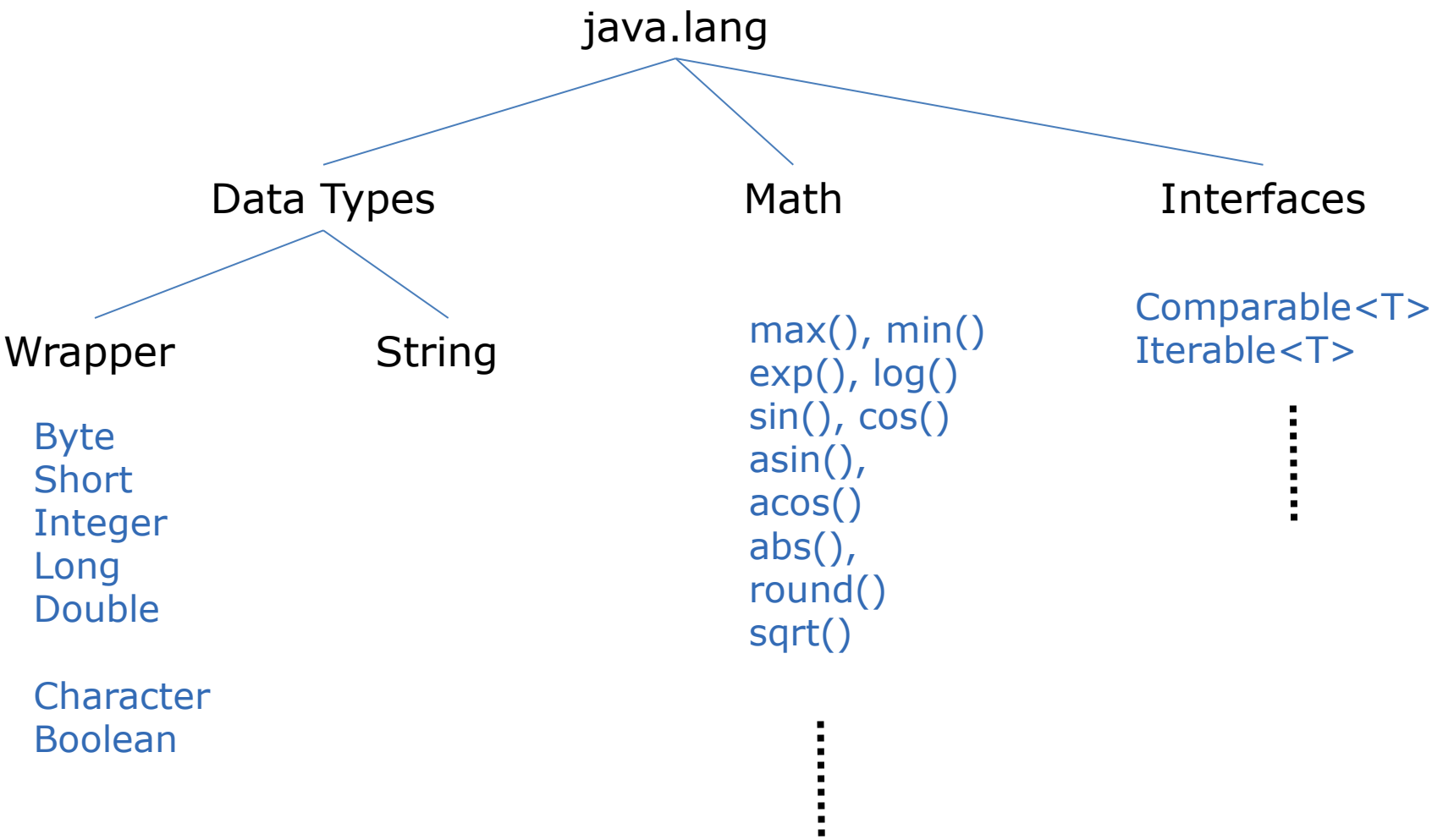
包名	类名
mygame.server	Server
mygame.shared	Utilities
mygame.client	Client

1）每个类的源代码为实现分包应该加入什么语句？

2）应该创建怎样的文件夹结构？

3）你认为每个类的源代码中还需做出什么调整吗？

2. 基础类


java.lang
Data Types
Math
Interfaces
Wrapper
String
Byte
Short
Integer
Long
Double
Character
Boolean
max(), min()
exp(), log()
sin(), cos()
asin(),
acos()
abs(),
round()
sqrt()
Comparable<T>
Iterable<T>

❑ java.lang.Object 类

所有类的祖先

仅有一个默认构造方法

```
public Object(){} // 方法体为空
```

主要成员方法：

equals(Object obj)：判断是否为同一对象

toString()：返回对象的字符串表示， “类名@对象的十六进制哈希码”

2. 基础类

❑ 包装类

基本类型	对应包装类
boolean	Boolean
byte	Byte
char	Character
short	Short
int	Integer
long	Long
float	Float
double	Double

包装类的特点：

1）所有的包装类都是final类型，不能创建子类

2）包装类是不可变类，不可强制转换

❑ 包装类的构造方法

所有的包装类都可以以它对应的基本数据类型为参数来构造实例

```
Boolean bln = new Boolean(true);
Character c = new Character('c');
Short s = new Short((short)1);
Integer I = new Integer(1);
```

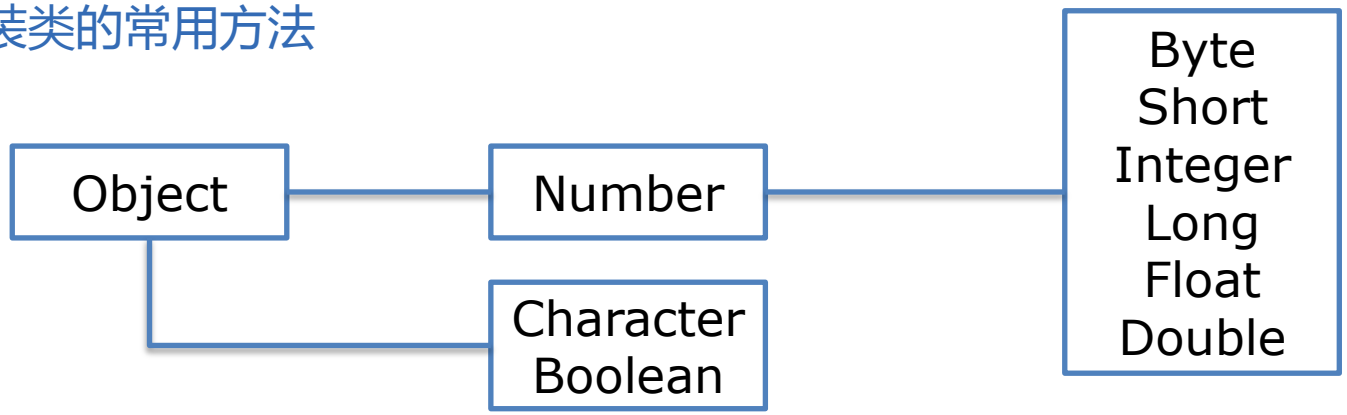
除了Character类外，都可以以一个字符串为参数来构造实例

```
Integer I = new Integer("123");
Double d = new Double("123.45D");
Float f = new Float("1.23F");
```

Boolean类的构造方法的参数为String时，仅当内容为“true”时为真（不考虑大小写）

```
Boolean("TrUe")
Boolean("FalSe")
```

❑ 包装类的常用方法



1) 包装类覆盖了Object类的toString()方法，打印时返回对应字符串值

2) 数值类都有valueOf(String s)静态方法，用于创建包装类对象

```
Integer i = Integer.valueOf("123");
Integer i = Integer.valueOf("abc");
```

3) 数值类都有parseXXX(String s)静态方法，用于将字符串转为对应的基本数据类型

```
Double d = Double.parseDouble("123");
```

❑ 包装类的自动转换 - 装箱、拆箱

```
// JDK5以后可行

Integer a = 3;
int b = new Integer(4);

List<Integer> list = new ArrayList<Integer>();
list.add(3);
int I = list.get(0);
```

2. 基础类

❑ Math类

Math是final类

有两个静态常量：E（自然对数）、PI（圆周率）

Math类不能被实例化

abs()
ceil()
floor()
max()
min()
round()

random()

sin()
cos()
tan()
asin()
acos()
exp()
pow()
sqrt()

3. 工具类

❑ Random类

public class Random
extends Object
implements Serializable

- 此类用了48-bit 的seed和线性同余公式，来产生伪随机数序列;
  (参看Donald Knuth, The Art of Computer Programming, Volume 2, Section 3.2.1.)
- 很多情形下 *double Math.random()* 更简单;
- java.util.Random 的实例是线程安全的;
- java.util.Random 的实例并未加密，如有需要则使用*SecureRandom*类。

3. 工具类

❑ Random类

public class Random
extends Object
implements Serializable

public synonronized void setSeed(long seed)	该方法是设定基值seed
public int nextInt()	该方法产生一个正或负整型随机数
public int nextInt(int n)	该方法产生一个不大于n的整型随机数
public long nextLong()	该方法是产生一个正或负long型随机数
public float nextFloat()	该方法是产生一个正或负Float型随机数
public double nextDouble()	该方法是产生一个正或负Double型随机数

练习2

```
Random rand = new Random();
int num = rand.nextInt(n);
```

请生成10个不超过100的随机整数。

```
public class RandomTest {
        public static void main(String[] args){
                java.util.Random rand = new java.util.Random();
              for(int i=0;i<10;i++){
                  System.out.println(rand.nextInt(100));
              }

         }
}
```

3. 工具类

- 日期类

java.util.Date

```
Date date = new Date()
date.getTime();
```

返回1970年1月1日零点距今毫秒数

java.util.Calendar

```
Calendar cal = Calendar.getInstance();
cal.set(Calendar.YEAR, 2017);
```

java.text.DateFormat

```
SimpleDateFormat dFormat = new SimpleDateFormat(“EEEE/MMMM/dd/yyyy”);
System.out.println(dFormat.format(date));
```

3. 工具类

❑ JDK8 日期时间类

java.time

- LocalDate
- LocalTime
- LocalDateTime

```
LocalDate today = LocalDate.now();

LocalDate firstDay2016 = LocalDate.of(2016,Month.JANURARY,1);

LocalDate todayParis = LocalDate.now(ZoneId.of(“Europe/Paris)”);

LocalDate dateFromBase = LocalDate.ofEpochDay(365);
```

练习3

如何计算两个日期之间相差的天数。

```
Date d1 = dateFormat.parse(“2017-03-11”);
Date d2 = dateFormat.parse(“2017-04-21”);

long p = d2.getTime() - d1.getTime();
return p/(1000*60*60*24);
```

3. 工具类

❑ BigXXX 类

java.math.BigInteger

如果整数的值比Long.MAX_VALUE还大，则需使用BigInteger

```
BigInteger a = new BigInteger(new Long
(Long.MAX_VALUE).toString());

BigInteger b = a.add(new BigInteger("1"));

System.out.println("a = "+a);
System.out.println("b = "+b);
```

java.math.BigDecimal

总结

包

Java类库

java.lang

- Object类
- 包装类
- Math类

java.util

- Random类
- 日期类
- BigXXX类

下节预告

Java IO