Assessment

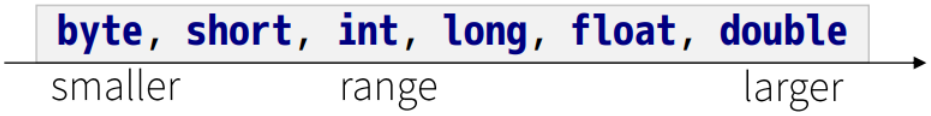
- Coursework 60%
  - Assignments 10% = 5% * 2
  - Project 25%
  - Quiz[1] (closed book) 25%

- Final Exam (closed book) 40%

class：首字母大写
常数：全大写

```
Scanner s = new Scanner(System.in);
int v = s.nextInt();
...nextLine()
```

Numeric Type Conversion (Cont'd)



- Implicit

```
double d = 3;    // type widening
```

- Explicit (casting)

```
int i = (int)3.5;  // type narrowing
int i = 5 / 2.0;   // error!
```

- Attention
  - Precision vs. Range

```
int x = 1234567890;
float y = x;
int z = (int) y;
// z is 1234567936
```

  - Augmented assignment

```
short x = 2;
x = x + 1.1; // error!
x += 1.1;    // OK!
// x = (short)(x + 1.1)
```

没明白

```
>>> 无符号位移
```

Java Operator Precedence Table

Operator	Description	Level	Associativity

`[]`
`.`
`()`
`++`
`--`
	access array element
	access object member
	invoke a method
	post-increment
	post-decrement
		1	left to right

`++`
`--`
`+`
`-`
`!`
`~`
	pre-increment
	pre-decrement
	unary plus
	unary minus
	logical NOT
	bitwise NOT
		2	right to left

`()` `new`	cast object creation	3	right to left

`*`
`/`
`%`
	multiplicative	4	left to right

记不住

Java Operator Precedence Table (Cont'd)

Operator	Description	Level	Associativity
`+ -` `+`	additive string concatenation	5	left to right
`<< >>` `>>>`	shift	6	left to right
`< <= > >=` `instanceof`	relational type comparison	7	left to right
`==` `!=`	equality	8	left to right
`&`	bitwise AND	9	left to right
`^`	bitwise XOR	10	left to right
`|`	bitwise OR	11	left to right
`&&`	conditional AND	12	left to right
`||`	conditional OR	13	left to right
`?:`	conditional	14	right to left

`= += -= *= /= %=`
`&= ^= |= <<= >>=`
`>>>=`
	assignment	15	right to left

How Are They Evaluated?

- Let

```
int a = 1, b = 2, c = 3;
```

what are the values of a, b, and c after executing the following statement?

```
a = b += c = a + 5;
```

- When in doubt, use parentheses!

搞不清楚

field, method -> instance member
reference 类名 变量创建

primitive type 原始的 / 其他的都是 reference type（比较的地址）
赋值（复制地址
receiver. 方法()
Invocation 调用

- 类里可以 this. 也可以直接 field 名字
- hidden / shadowed (field 优先级低，只能 this
- 局部可以任意
- `new 对象()` 创建
- Constructors 构造函数
- 自己调用其他构造函数 `this()`

初始化

- Instance Initializers (or instance initialization blocks)
- 可以直接给变量赋值
- 除了变量按顺序调用
  < Method signature

static

每个对象只有一个
没 dis

Initialization Order

```
class Hero{
    String name;

    static Hero heroA = new Hero("M. J.", 2);

    static int x = 1;

    Hero(String name, int strength){
        this.name = name;
        System.out.print(x+" ");
    }

    public static void main(String[] args){
        Hero beauty = new Hero("Beauty", 20);
    }
}
```

```
0 1
```

调用栈

Access control

- public
- private：同一个 class
- package -
  每个 file 只能一个 class

Modifier	Class Itself	Other Class (same pkg)	Subclass (same pkg)	Subclass (diff pkg)	World
public	Y	Y	Y	Y	Y
protected	Y	Y	Y	Y	N
(no modifier)	Y	Y	Y	N	N
private	Y	N	N	N	N

default：sub 不同 package 不行，保护是 world 不行

protected和default唯一区别就是不同包的subclass pro可以defa寄了
Singleton：一个实例
private static Logger instance;
reference 没事，就是地址不能变而已。
final 类 c++ 的 const（赋值一次

package 命名：全小写，域名 rev

How Does Java Virtual Machine Locate a Class? (1)

- The default class loader searches for classes using a list of paths defined in “classpath”
  - A classpath is just a group of paths

```
c:\jdk\src;d:\mylib\src; // On Windows.
```

  - Can be set in an environment variable or passed to Java tools as an argument

```
c:\> set classpath=c:\jdk\lib;d:\mylib1;
```

```
c:\> java -classpath c:\jdk\lib;d:\mylib2 <other parameters>
```

    - classpath as command line argument will overwrite the value set in the environment variable
    - The default value for classpath includes only the current working directory (.).

有点难了
用 full name 或者 import。default 里的别人不能用
import class 重名会 error

Lecture 5

```
enum X{
        A(Para);
        int t;
        X(int x) {
                t = x;
        }
}
```

```
// 声明细节

// switch 反常不用类名

str.trim() // 去掉两遍空格
str.substring(l, r) [l, r)
str.indexOf(s2 / 's', a)
str.lastIndexOf(s2 / 'v', b) // <= 3


String.valueOf()
Integer.parseInt()
Double.parseDouble()
SB append, toString()
//
```

Enum

可以当 class

Wrapper Classes

Autoboxing and Unboxing
box 封装到包装类，不然是反过来的

Lecture 6

polymorphically 多态

只能继承他找得到的（同包 / public
`extends`
只有 member。构造函数，静态初始化器，实例初始化器

private 还是一部分，但是可能没法昂文

Overriding	Overloading
Method names should be the same	
Overrides a method from superclass	Overloads a *member method*
With the same signature	With a different signature
Is a runtime concept	Is a compile-time concept
Return type must be the same or a subtype	Return type does not matter
Accessibility must be more permissive	Accessibility does not matter

注意 rid 是 runtime
注意一手充要条件：权限可以边宽松：
public->protected->(default)->private
private不能被override
运行时 - 编译问题
return也是subtype

What Are Inherited? (2)

			p.A	p.B	p.C extends p.A	q.D extends p.A	q.E
p.A	Field (static or not)	public	Y	Y	Y	Y	Y
		protected	Y	Y	Y	Y	N
		(default)	Y	Y	Y	N	N
		private	Y	N	N	N	N
	Method (static or not)	public	Y	Y	Y	Y	Y
		protected	Y	Y	Y	Y	N
		(default)	Y	Y	Y	N	N
		private	Y	N	N	N	N

- p, q: packages
- A, B, C, D, E: classes

注意除了私密同胞都能访问，不同包只能公共，除非继承了又保护
保护外包都改不了，继承了可以

继承以后第一句必须是父类构造函数（super（如果没有明确定义就是五参数输入

从高到低初始化

Class Instance Creation

- Creating a new class instance
  - Storage allocation for all the fields (declared in the current or super-classes)
  - All instance variables initialized to default values
  - Decide which constructors are invoked in each superclass
  - Do the following for each class, from the top superclass to the bottom subclass
    - Execute the instance initializer and instance variable initializer in their textual order
    - Execute the corresponding constructor(s)

```
class A {
  A(String s){
    System.out.println(s);
  }
}
class E {
  { a2 = new A("a2"); }
  A a1 = new A("a1");
  A a2;
}
class F extends E {
  A a3 = new A("a31");
  { a3 = new A("a32"); }
  A a4;
  F()
    { this(5); }
  F(int i)
    { a4 = new A("a4"); }
}
// what's the output?
F f = new F();
```

```
a2
a1
a31
a32
a4
```

Class Instance Creation 到根的链从上往下推

Class Initialization：第一次用懒惰vis没访问的向下腿
没被继承不能覆盖（比如private
覆盖：需要signature一样（参数一样 return还得一样
overloading

Polymorphic 只要是他的儿子都可以多态?
T_C
static type 定义
dynamic type refer的变量名

反过来不行。

```
instanceof
```

Dynamic Binding (Late Binding)

Name Binding: Static or Dynamic （我们是动态）
静态绑定的field

Static Binding for Static Member Access(static一直是定义的那个type)
Dynamic Binding for Overridden Methods
Static Binding for Calls on super
Resolving Methods at Compile Time（运行时间check
ambiguous call 找不准确，没有恰好对的。

override：权限更严格，还得是返回值

Lecture 7

默认 extends object：

```
toString
equals
HashSet<>()
h.contains(o)
h.add(..)
```

Abstract 抽象 不实现
有一个抽象方法，就成为了抽象类。
静态 private instance，够咱含税都不能抽象
抽象类也不能实例化
具体 override 抽象
构造系统

interface（empty body）

interface (key word)
Instance（public + abstract）
Static: public static final (public concrete)

The Cloneable Interfaces
clone

implements Cloneable

HashSet

initial capacity
load factor

Midterm

1-5
相同包是运行时错误。
、

Lecture 8 UML

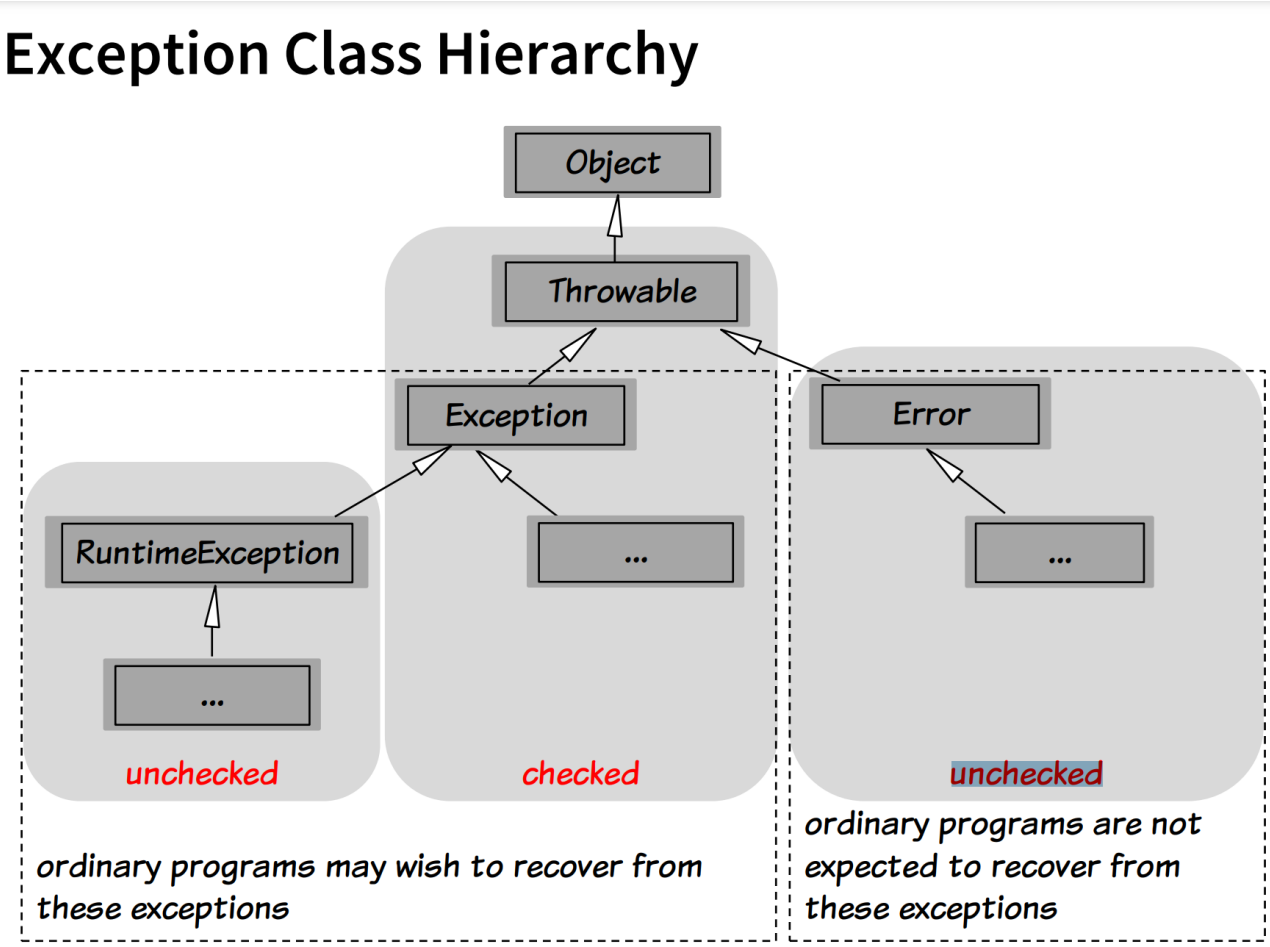
- Specification: 足够简单理解
- Visualization: graphically 表示
- Construction: 足够智慧可以重建代码
- Documentation: 广泛流传 其他开发者理解

Class

- Name /Attributes / Operations
- - private + public # protected / derived / _ staitc (球球，别背了)
-
- _
- dependency A 依赖 B (改 B， A 会变) $A \rightarrow B$（虚箭头）
- Associations 用什么函数箭头指
- Association Multiplicity 包含多少：实现箭头三角 （x..y 或 x 或 `*` (任意)）
- Special Associations (Has-A Relation)：
  - Aggregation: 空菱形弱 包含 任意个 *
    - Composition: （strong aggregation）：实菱形包含 n 哥（不能删
- Navigability of Association 能访问对面就指向对面一个箭头
- 抽象函数：{abstract}
- interface 函数 `<<interfaces>>`
- 继承：虚菱形 实箭头 儿子指父亲
  UML Object Diagrams
- 无限：x..y

Lecture 9 Exception Handling

nearest dynamically enclosing catch clause (NDECC)



Exception：希望回复 (RE check不到) 只
Error 不用恢复
Resumption Model of Exception Handling

可以用杠隔开

Checked - 除了Runtime Exception以外的 Exception

try (要关掉的局部变量) catch finally 都可以不要
一般的 try 后面至少一个

Generics

Generic Class

Generic Methods

Object 就不用 extends 了，直接？就好

```
<? extends ... super ...>
```

Wildcard 通配符
extends 上界
super 下界

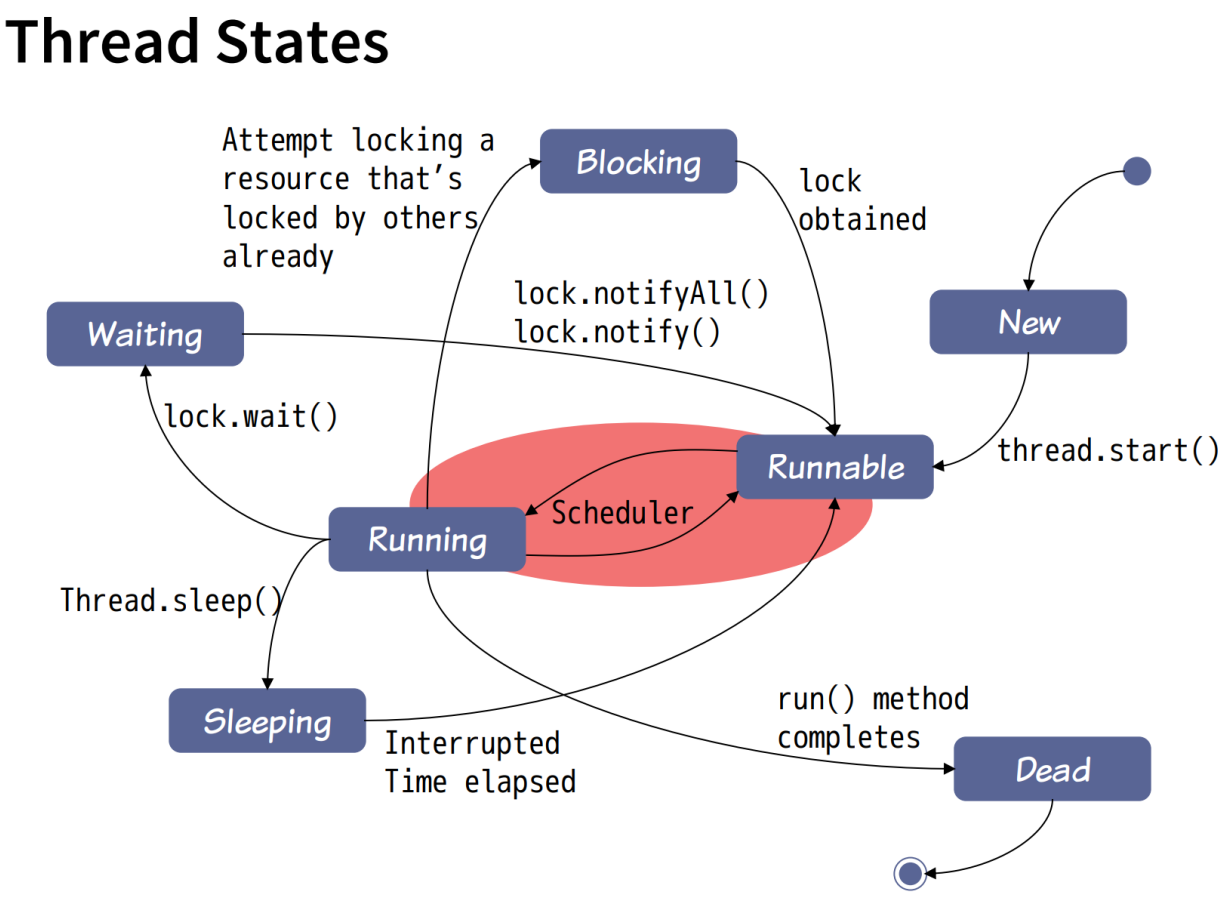
Lecture 10

- Processes 进程 不同空间
- Thread 线程（轻量进程）分享地址空间 ☆ Java

join 等待结束。

共享 = critical region
Mutex (mutual exlusion object)
Monitor = Mutex + Condition variable
lock / unlock

用 synchronized(变量名, this (函数前加等价)) 进入同步状态（monitorenter / exit)



守护线程 Daemon

Lecture 11

- AWT (Abstract Windowing Toolkit): simple / plateform-specific bugs.

- JavaFX: destop applications / rich internet applicaitions (RIA) 很多设备。要取代 AWT 和 Swing。但 Swing 更好解释面向对象

Swing

1. Pluggable look and feel
2. 无限次 undo redo

组件

- 顶层 ① Window: JFrame / JDialog ② Panel: Japplet 选一个主要的。
  - 有一个 content pane。 要显示得在里面 add。只能 add 一次。
  - 如果要 Content Pane 是 JComponent 要自己 new 一个然后 set
- Intermediate: JPanel / JtabbedPane
- 原子组件：JButton / JTextFiled

Event-Driven

Event source object (源头) -> event object (捕捉 封装信息) -> Event handler object (你需要创建的处理对象)

```
java.util.EventObject // 大
.getSource()
```

每个组件能触发哪些事件：

- 都能触发的
- ActionListener: 选择器 button 点的时候
- CaretListener：打字光标的时候

Nested Classes

- Static Nested Class: 使其不依赖外面的实例。
- Inner Classes: 外对象.new Inner Class()
- Local Classes：能访问外面的 member 和没改过的变量 / final
- 匿名 class：创建的时候直接花括号改改方法 owo

Lambda

(参数) -> {} / exp 写点什么！

Lecture 12 Basic IO

不考，已删掉。