Introduction & Java Review

EE 564

Lecture 1

Daqing Hou, Winter 2007

Course Overview

- Personnel introduction

- Course plan:
  theory: OO design 1 month
  practice 1: eclipse ~1.5 months
  practice 2: Java EE ~1.5 months

- Plan to devote 10 hours to this course per week

Java Review

- Variables and objects
- Program structure
- Type checking
- Dispatching

Variables and objects

- Variable: name + storage address
- Storage stores values
- 3 kinds of value
  primitives, objects, references
- Stack, activation record, and heap
- Object: creation, sharing, mutability
- Method invocation

Variables and objects: ex

```
int i=6;
int j; // uninitialized
int [] a ={1,3,5,7,9};// create a 5-element array
int [] b = new int[3];
String s = “abcdef”; // create a new string
String t = null;
```

Sharing and mutability

```
int i=6;
int j; // uninitialized
int [] a ={1,3,5,7,9};// create a 5-element array
int [] b = new int[3];
String s = “abcdef”; // create a new string
String t = null;
```

```
j = i;
b = a;
t = s;
```

```
t = t+”g”;
b[0] = i+1;
a[1]=(a[0]==7)?5:3;
```

Method invocation

```
Class Arrays {
 public static void multiplies(int[]a, int m){
    if (null==a) return;
    for (int i=0; i<a.length; ++i){
        a[i] = a[i]*m;
    }
 }
}

int [] b = {1, 3, 5, 7, 9};
Arrays.multiplies(b, 2); // pass by value
```

Garbage collection

```
Class Stack{
 Object[] data;
 int p;
 Stack(int size) {data = new Object[size];p=0;}
 void push(Object e) {data[p++]=e;} // may throw
   IndexOutofBoundsException
 Object pop() {return data[--p];}…
}

Stack stack = new Stack(10 000 000);
for (int i=0; i<10 000 000; ++i)
{stack.push(new Integer(i));}
for (int i=0; i<10 000 000; ++i)
{stack.pop();}
```

Program structure

- Classes and interfaces
  - standalone operations
  - nested classes
- Packages
  - naming; fully-qualified names
  - encapsulation via access control
- Access control
  - for members
  - for top-level classes and interfaces

Type checking

- Type checking rules out type errors
- Declaration types, apparent types, and actual types
- Type hierarchy, subtyping, typing rules
- Conversions and overloading
- But, type checking cannot rule out all programming errors
  - "`(struct S *)(0x1)->m`" compiles in C

Declaration types, apparent types, and actual types

```
int x, y;
static int gcd(int m, int n);
int s = gcd(x,y);
long t = gcd(x,y);

String s = “abc”;
Object o = s;
```

Conversions and overloading

- Explicit conversions, aka casting
  (T)d
- Implicit conversions (widening)
  - e.g. int->long, float->long
- Overloading
  - operator overloading, e.g., +
  - method name overloading

Type hierarchy & typing rules

- E.g.: all classes are subtypes of java.lang.Object
  - defines equals(), hashCode(), toString(), etc
- Subtype relation
  - reflexive
  - transitive
  - antisymmetric
- Typing rules. E.g., for assignment: `v=e;` T(e) must be a subtype of T(v)

Overloading resolution: ex1

```
static int comp(int,long)  // def 1
static int comp(long,int)  // def 2
static int comp(long,long) // def 3

int x; long y; float z;
C.comp(x,y)
```

"Most-specific" rule:

- methods requiring least conversions win
- tie as compilation error

```
C.comp(z,z)
C.comp(x,x)
```

Overloading resolution: ex2

```
static void comp(Sup,int)    // def 1
static void comp(Sub,long)   // def 2

Sup is supertype of Sub

Sub e; int i;

C.comp(e,i);
```

Dispatching

- When a method is called on an object, it is essential to invoke the code provided by the object.
- Dispatch vector:
  a runtime mechanism that stores list of methods defined by object's class.

Dispatching

```
String t="ab";
Object o = t+"c";
Object r = "abc";
boolean b = o.equals(r);
```

What is the value of b and explain why?

Exercise 1

```
Object o = “abc”;
boolean b = o.equals(“a, b, c”);
char c = o.charAt(1);
Object o2 = b;
String s = o;
String t = (String)o;
c=t.charAt(1);
c=t.charAt(3);
```

Exercise 2

```
char[] a = {'a','b','c'};
Object o = "abc";
String t="ab";
String w = t+"c";
String u=w;
boolean b0 = o.equals(a);
boolean b1 = o.equals(t);
boolean b2 = o.equals(w);
boolean b3 = (o==w);
boolean b4 = (u==w);
```

Exercise 3

```
void m(Object o, long x, long y) // def 1
void m(Object o, int x, long y)  // def 2
void m(String s, int x, long y)  // def 3
void m(String s, long x, int y)  // def 4

Object u; String v; int a; long b;

m(v,a,b);
m(v,a,a);
m(v,b,a);
m(v,b,b);
m(u,b,b);
m(u,a,a);
```

Summary

- Variables and Objects
- Program structures
- Type checking
- Dispatching

Assignment 1

- Exercises 2.1, 2.2, 2.3. (1%)
- Small programming 1 (2%)
- Small programming 2 (2%)

- You are required to use Eclipse. Go to www.eclipse .org and download Eclipse. If you have any difficulty, ask.
- Discussion is encouraged, but verbatim copies are not allowed.
- Due: Jan. 22 in class. Email me your solution for the two small programming tasks.

Assignment 1

Programming 1 (2%)

- write a routine sum(int a[]) which modifies its argument a so that when it returns, each element a[i] contains the sum of the values of a[i], a[i+1], ..., a[size-1] as of the time of the call, where size is the length of the array a.
- Write a main program to print out the content of an array before and after sum is called.

Assignment 1

Programming 2 (2%)

- write a routine lex(T1 input, T2 output) that is capable of splitting a text into words and then writing them out to an output device, one word per line. It is your freedom to define what constitutes a word. But your routine should accept a variety of data types, like a Java string, a file, or a networking connection, as arguments.
- Write a main program to demonstrate the different uses of lex().
- Your need to do a little research to decide the proper types for T1 and T2. Hint: google and study java.io.Reader, java.io.Writer, and their subtypes.