ECE 462
Object-Oriented Programming using C++ and Java

Lecture 8

Yung-Hsiang Lu

yunglu@purdue.edu

Java List

```
//ListOps.java
import java.util.*;
class ListOps {
    public static void main( String[] args )
    {
        List <String> animals = new ArrayList <String>();
        animals.add( "cheetah" );
        animals.add( "lion" );
        animals.add( "cat" );
        animals.add( "fox" );
        animals.add( "cat" );          //duplicate cat
        System.out.println( animals );
        //cheetah, lion, cat, fox, cat
```

different from textbook for Java 1.5

```
        animals.remove( "lion" );
        System.out.println( animals );  //cheetah, cat, fox, cat
        animals.add( 0, "lion" );       // overload, two arguments
        System.out.println( animals );  //lion, cheetah, cat, fox, cat
        animals.add( 3, "racoon" );
        System.out.println( animals );  //lion, cheetah, cat, racoon, fox, cat
        animals.remove(3);
        System.out.println( animals );  //lion, cheetah, cat, fox, cat
        ListIterator iter = animals.listIterator();
        while ( iter.hasNext() ) {
            System.out.println( iter.next()  );
        }
    }
}
```

Java Set (no duplicate element)

```
//SetOps.java
import java.util.*;
class SetOps {
    public static void main( String[] args )
    {
        Set <String> animals = new TreeSet<String> ();          //(A)
        animals.add( "cheetah" );                               //(B)
        animals.add( "lion" );                                  //(C)
        animals.add( "cat" );                                   //(D)
        animals.add( "elephant" );                              //(E)
        animals.add( "cat" );              // duplicate cat     //(F)
```

```
        System.out.println( animals );                    //(G)
                                // cat cheetah elephant lion
          System.out.println( animals.size() );       // 4          //(H)
          animals.remove( "lion" );                    //(I}
          System.out.println( animals );  // cat cheetah elephant   //(J)

          Iterator iter = animals.iterator();           //(K)
          while ( iter.hasNext() )                      //(L)
             System.out.println( iter.next() );         //(M)
                                // cat cheetah elephant
    }
}
```

Java Map (Hash Table)

- **array: integer → element**
- **map: key (object, integer, string ...) → value**
  example: name → phone number,
  student ID → department
  city name → zip code
  Keys are not necessarily continuous.

```
//MapHist.java
import java.io.*;
import java.util.*;
class WordHistogram {
    public static void main (String args[]) throws IOException
    {
        Map <String, Integer> histogram =
                        new TreeMap<String, Integer>();
        String allChars = getAllChars( args[0] );
        StringTokenizer st = new StringTokenizer( allChars );
```

```
    while ( st.hasMoreTokens() ) {
        String word = st.nextToken();
        Integer count = (Integer) histogram.get( word );
        histogram.put( word, ( count==null ? new Integer(1)
              : new Integer( count.intValue() + 1 ) ) );
    }
   System.out.println( "Total number of DISTINCT words: "
                        + histogram.size() );
    System.out.println( histogram );
}
```

```
static String getAllChars( String filename ) throws IOException {
        String str = "";
        int ch;
        Reader input = new FileReader( filename );
        while ( ( ch = input.read() ) != -1 )
            str += (char) ch;
        input.close();
        return str;
    }
}
```

Container Class (code reuse)

- Many programs need "containers" to store information. Examples of containers include vector, list, stack, queue, map, and set.
- A container needs to be able to hold items of different types (i.e. classes). Examples
  - list of strings, integers, floating points, student objects
  - queues of customer objects, car objects
  - maps: name $\rightarrow$ address, student ID $\rightarrow$ name, course title $\rightarrow$ classroom
- C++ standard template library (STL) and Java container classes provide such functionality.

Select Container Class

- All container classes have internal memory management, automatic allocation or release.
- random or sequential accesses
- allow unique or duplicate items
- O(1) or O(N) for array-like access (using [index])
- efficient insert / delete
  - front
  - end
  - middle
- Java containers cannot store primitive types (int, char, float ...)

Efficiency

operation	vector	deque	list
array-like access	O(1)	O(1)	O(N)
insert/delete at front	O(N)	O(1)+	O(1)
insert/delete at end	O(1)+	O(1)+	O(1)
insert/delete in middle	O(N)	O(N)	O(1)

N: current number of items

```
import java.io.*;
import java.util.*;
class Student {
    private int s_ID;
    private String s_name;
    private String s_department;
    public Student(int id, String nam, String dept) {
        s_ID = id;          s_name = nam;              s_department = dept;
    }
   public String toString() {
        String rtv = "\nID = " + s_ID + "\n\tname = " + s_name +
            "\n\tdepartment = " + s_department + "\n";
        return rtv;
    }
}
```

```
class VectorClass {
 public static void main( String[] args )
 {
    Vector<Student> stdVec = new Vector<Student>();
    Student s1 = new Student(109032, "James", "Math");
    Student s2 = new Student(100075, "Amy", "ECE");
    stdVec.addElement(s1);
    stdVec.addElement(s2);
    System.out.println(stdVec);
 }
}
```

```
class MapClass {
    public static void main( String[] args )
    {
        Map<Student, String> stdMap =
            new HashMap<Student, String>();
        Student s1 = new Student(109032, "James", "Math");
        Student s2 = new Student(100075, "Amy", "ECE");
        Student s3 = new Student(200069, "David", "CS");
        stdMap.put(s1, "Apartment 1, Lafayette");
        stdMap.put(s2, "Apartment 2, West Lafayette");
        System.out.println(stdMap.get(s1));
        System.out.println(stdMap.get(s3));
    }
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <string>
using namespace std;
class Student
{
private:
  int s_ID;
  string s_name;
  string s_department;
public:
  Student(int id, string nam, string dept):
    s_ID(id), s_name(nam), s_department(dept) { }
  friend ostream & operator << (ostream & os, const Student & std);
};
```

```
ostream & operator << (ostream & os, const Student & std)
{
   os << endl << "ID = " << std.s_ID << endl
      << "\tname = " << std.s_name << endl
      << "\tdepartment = " << std.s_department << endl;
    return os;
}

int main()
{
    vector<Student> vec;
    Student s1(7009, "Mary", "ECE");
    Student s2(7184, "Tom", "Physics");
    Student s3(6553, "Jennifer", "Chemistry");
```

```
    vec.push_back( s1 );
    vec.push_back( s3 );
    vec.push_back( s2 );

    vector<Student>::iterator p = vec.begin();
    while (p != vec.end())
      {
        cout << (*p);
        p++;
      }
}
```

Copy Constructor

```
class NameOfClass
{
    NameOfClass(const NameOfClass & origobj)
    {
        // a constructor with special syntax
        // If a user does not provide a copy constructor
        // C++ compiler automatically creates one
    }
}
```

ECE 462
Object-Oriented Programming using C++ and Java

Lecture 9

Yung-Hsiang Lu

yunglu@purdue.edu

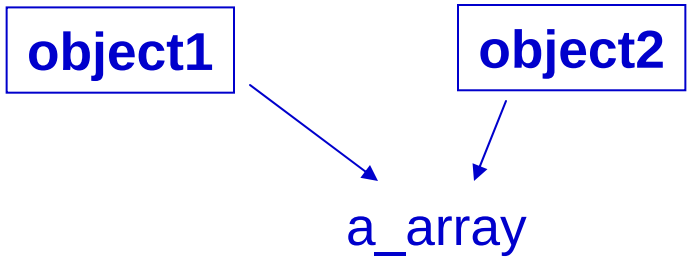
C++ Constructor, No-arg Constructor, Copy Constructor, =

- Constructors can be overloaded: same name with different parameter list (number of parameters and their types). Destructor cannot be overloaded.
- C++ compiler automatically creates
  - no-arg constructor, if no constructor is provided by a programmer. All attributes are initialized to their default values, or by their no-arg constructors.
  - copy constructor, if no copy constructor is provided.
  - operator = (assignment), if it is not provided

Shallow or Deep Copy

The default copy constructor and operator = by C++ compiler performs “shallow” copy: only copy values. This causes problems if an object contains pointers.

```
class AClassName
{
      private:
         int *a_array;
      public:
         AClassName(...)
         { a_array = new int[ARRAY_SIZE]; }
}
```



Sharing Memory among Objects

- use a static attribute to count the number of objects: count ++ in constructor, count -- in destructor
- memory location also a static member
- if (count == 1) allocate memory
- if (count == 0) release memory

Lab 5: Comparing the Performance of C++ and Java Container Classes

measure the time

- C++ and Java (linked) lists of strings
  - push_back 1 million strings
  - push_front 1 million strings
- C++ and Java set of Student objects
  - insert 1 million distinct objects
  - insert 1 million duplicate objects

```
/* ::::::::::::::
listperformance.cpp
:::::::::::::: */
#include <iostream>
#include <string>
#include <list>
#include <sys/time.h>
using namespace std;
void print (list<string> lab5)
{
  list<string>::iterator it = lab5.begin();
  while (it != lab5.end())
    {
      cout << (*it) << endl;
      it ++;
    }
}
int main(int argc, char * argv[])
{
int NUMBER_STRING = 10;
 if (argc > 1)
   { NUMBER_STRING = atoi(argv[1]); }
 if (NUMBER_STRING < 10) {
     NUMBER_STRING = 10; }
cout << "NUMBER_STRING = " <<
     NUMBER_STRING << endl;
 list<string> lab5list;
 struct timeval tp1;
 struct timeval tp2;
 char buffer[256];
 string element;
 gettimeofday(& tp1, NULL);
 for (int itercnt = 0; itercnt <
     NUMBER_STRING; itercnt ++)
  {
     sprintf(buffer, "stringelement%0d",
     itercnt);
     element = buffer;
     lab5list.push_back(element);
  }
```

```
gettimeofday(& tp2, NULL);
 cout << "push_back time = " <<
   1e-6 * (tp2.tv_usec - tp1.tv_usec) +
   (tp2.tv_sec - tp1.tv_sec) << endl;
 // print(lab5list);
gettimeofday(& tp1, NULL);
 for (int itercnt = 0; itercnt <
     NUMBER_STRING; itercnt ++)
   {
     sprintf(buffer, "stringelement%0d",
     itercnt);
     element = buffer;
     lab5list.push_front(element);
   }
   gettimeofday(& tp2, NULL);
   cout << "push_front time = " <<
     1e-6 * (tp2.tv_usec - tp1.tv_usec) +
     (tp2.tv_sec - tp1.tv_sec) << endl;
   // print(lab5list);
   return 0;
}
```

```
/* ::::::::::::::
listperformance.java
:::::::::::::: */
import java.util.*;
// May need to give more memory for Java
// try "java -Xms256m -Xmx512m"
//  initial heap heap size ({\tt -Xms}) to be
      256MB and the maximum
// heap size ({\tt -Xmx}) to be 512MB.
class listperformance
{
    public static void main( String[] args )
    {
      int NUMBER_STRING = 10;
      if (args.length > 0)
          { NUMBER_STRING =
      Integer.parseInt(args[0]); }
      if (NUMBER_STRING < 10) {
      NUMBER_STRING = 10; }
System.out.println("NUMBER_STRIN
G = " + NUMBER_STRING);
List <String>lab5list = new
          LinkedList<String>();
String element;
long time1 =
System.currentTimeMillis();
for (int itercnt = 0; itercnt <
NUMBER_STRING; itercnt ++)
    {
          element = "stringelement" +
itercnt;
          lab5list.add(element);
    }
long time2 =
System.currentTimeMillis();
System.out.println("add time = " +
                              1e-3 * (time2 -
time1));
// System.out.println(lab5list);
```

```
    time1 = System.currentTimeMillis();
    for (int itercnt = 0; itercnt <
    NUMBER_STRING; itercnt ++)
       {
            element = "stringelement" +
    itercnt;
            lab5list.add(0, element);
       }
    time2 = System.currentTimeMillis();
    System.out.println("add(0, time = " +
                          1e-3 * (time2 -
    time1));
    // System.out.println(lab5list);
   }
}
```

```
/* ::::::::::::::
setperformance.cpp
:::::::::::::: */
#include <iostream>
#include <string>
#include <set>
#include <sys/time.h>
using namespace std;
int callcount = 0;
class Student
{
private:
  int s_ID;
public:
  Student(int id): s_ID(id)
  { }
  void print(void) const
  { cout << "ID: " << s_ID << endl; }
```

```
 bool operator == ( const Student & std)
     const
  { callcount ++; return (s_ID == std.s_ID);
     }
  bool operator < ( const Student & std)
     const
  { callcount ++; return (s_ID < std.s_ID); }
};
void print(set<Student> lab5)
{
  set<Student>::iterator it = lab5.begin();
  while (it != lab5.end())
    {     (*it).print();     it ++;    }
}
int main(int argc, char * argv[])
{
  int NUMBER_STUDENT = 10;
  if (argc > 1)
    { NUMBER_STUDENT = atoi(argv[1]); }
if (NUMBER_STUDENT < 10) {
     NUMBER_STUDENT = 10; }
  cout << "NUMBER_STUDENT = " <<
     NUMBER_STUDENT << endl;
  set<Student> lab5set;
  struct timeval tp1;
  struct timeval tp2;
  gettimeofday(& tp1, NULL);
  for (int itercnt = 0; itercnt <
     NUMBER_STUDENT; itercnt ++)
   {
     Student std(itercnt);
     lab5set.insert(std);
   }
  gettimeofday(& tp2, NULL);
  cout << "add distinct time  = " <<
    1e-6 * (tp2.tv_usec - tp1.tv_usec) +
    (tp2.tv_sec - tp1.tv_sec) << " " <<
     callcount << endl;
  // print(lab5set);
```

```
 callcount = 0;
  gettimeofday(& tp1, NULL);
  for (int itercnt = 0; itercnt <
      NUMBER_STUDENT; itercnt ++)
    {
      Student std(itercnt);
      lab5set.insert(std);
    }
  gettimeofday(& tp2, NULL);
  cout << "add duplicate time = " <<
    1e-6 * (tp2.tv_usec - tp1.tv_usec) +
    (tp2.tv_sec - tp1.tv_sec) << " " <<
      callcount << endl;
  // print(lab5set);
  return 0;
}
```

```
/* ::::::::::::::
setperformance.java
:::::::::::::: */
import java.util.*;
class Student implements Comparable
{
    public static int callcount = 0;
    private int s_ID;
    public Student(int id) { s_ID = id; }
    public String toString()
    {
      String rtv = "ID: " + s_ID;
      return rtv;
    }
   public int compareTo(Object obj)
    {
      callcount ++;
      Student std = (Student) obj;
```

```
    if (std != null)
       {
            if (s_ID < std.s_ID) { return -1; }
            if (s_ID > std.s_ID) { return 1; }
            return 0;
       }
    else
       { throw new ClassCastException(); }
  }
};
class setperformance {
  public static void main( String[] args )
  {
     int NUMBER_STUDENT = 10;
     if (args.length > 0)
        { NUMBER_STUDENT =
    Integer.parseInt(args[0]); }
if (NUMBER_STUDENT < 10) {
        NUMBER_STUDENT = 10; }
System.out.println("NUMBER_STUDE
NT = " + NUMBER_STUDENT);
Set <Student> lab5set = new
TreeSet<Student>();
long time1 =
System.currentTimeMillis();
for (int itercnt = 0; itercnt <
NUMBER_STUDENT; itercnt ++)
    {
       Student std = new
Student(itercnt);
       lab5set.add(std);
    }
 long time2 =
System.currentTimeMillis();
 System.out.println("add distinct time
= " +  1e-3 * (time2 - time1) +    " " +
Student.callcount);
// System.out.println(lab5set);
```

```
    Student.callcount = 0;
    time1 = System.currentTimeMillis();
    for (int itercnt = 0; itercnt <
    NUMBER_STUDENT; itercnt ++)
        {
            Student std = new
    Student(itercnt);
            lab5set.add(std);
        }
    time2 = System.currentTimeMillis();
      System.out.println("add duplicate
    time = " +
                    1e-3 * (time2 - time1) +
                          " " +
    Student.callcount);
    // System.out.println(lab5set);
  }
}
```