

CLASSES, OBJECTS AND PACKAGES

15TH LECTURE

엄현상(Eom, Hyeonsang)
School of Computer Science and Engineering
Seoul National University

©COPYRIGHTS 2016 EOM, HYEONSANG ALL RIGHTS
RESERVED

Outline

- **Classes**
 - Creating New Data Types: class
 - Methods, Arguments and Return Values
- **Objects**
 - You Manipulate Objects Using References
 - Primitives
 - You Never Destroy Objects
- **Packages**
 - Using Other Components

Creating New Data Types: class

- Class keyword defines new data type

```
class ATypeName { /* class body goes here */ }
ATypeName a = new ATypeName();
```

- Fields

```
class DataOnly {
    int i;
    float f;
    boolean b;
}
```

- Each instance of **DataOnly** gets its own copy of the fields
- In a class, primitives get default values

Methods, Arguments and Return Values

- Methods: how you get things done in an object
- – Traditionally called “functions”
- – Can only be defined inside classes

```
ReturnType methodName(/* Argument list */) {  
    // Method body  
}
```

- Example method call:

```
int x = a.f(); // For object a
```

You Manipulate Objects Using References

```
String s; // Reference only
// Normal object creation:
String s = new String("asdf");
// Special string initialization:
String s = "asdf";
```

Primitives

- Built-in types: *not* object references, but variables on the stack like C.
 - **boolean, char** (Unicode), **byte, short, int, long, float, double**
- Same operations as C/C++, same syntax
- Size of each data type is machine independent!
- Portability & performance implications
- To create objects, wrapper classes are provided:
 - **Boolean, Character, Byte, Short, Integer, Long, Float, Double.**

```
char ch = 'x';  
Character c = new Character(ch);  
Or  
Character c = new Character('x');
```

Primitives Cont'd

Primitive type	Size	Minimum	Maximum	Wrapper type
boolean	—	—	—	Boolean
char	16-bit	Unicode 0	Unicode $2^{16}-1$	Character
byte	8-bit	-128	+127	Byte
short	16-bit	-2^{15}	$+2^{15}-1$	Short
int	32-bit	-2^{31}	$+2^{31}-1$	Integer
long	64-bit	-2^{63}	$+2^{63}-1$	Long
float	32-bit	IEEE754	IEEE754	Float
double	64-bit	IEEE754	IEEE754	Double
void	—	—	—	Void

You Never Destroy Objects

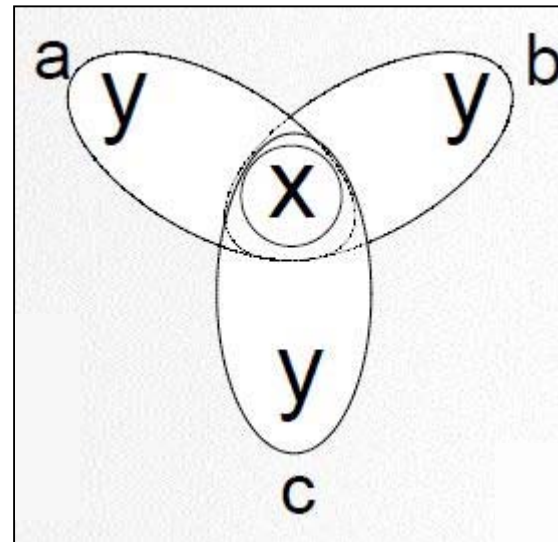
- Scope of objects

```
{ // <-- Beginning of scope  
String s = new String("a string");  
} // <-- End of scope  
// Reference has gone "out of scope"  
// but the object itself still exists
```


static Data (“class data”)

- Normally each object gets its own data
- What if you want only one piece of data shared between all objects of a class?

```
class WithStaticData {  
    static int x;  
    int y;  
}  
WithStaticData  
a = new WithStaticData(),  
b = new WithStaticData(),  
c = new WithStaticData();
```



Using Other Components

- Bring in a library of components using **import** keyword
- To specify particular element in library:
 - **import com.bruceeckel.utility.MyClass;**
- To specify entire library:
 - **import java.util.*;**

Package: the Library Unit

- Managing “name spaces”
 - Class members are already hidden inside class
 - Class names could clash
 - Need completely unique name even over the Internet
- Compilation units (**.java** files)
 - Name of **.java** file == name of single **public** class
 - Other non-**public** classes : not visible
 - Each class in file gets its own **.class** file
 - Program is a bunch of **.class** files (no **.obj** or **.lib**)

Creating a Library of Classes

- **package mypackage;**
 - **public** class is under the umbrella **mypackage**
 - Client programmer must **import mypackage.*;**
- Creating unique package names
 - Location on disk encoded into package name
 - Convention: first part of package name is Internet
 - domain name of class creator (reversed)
 - Java interpreter uses CLASSPATH environment
 - variable as starting point for search