Java Object Model

Part 1
Type

• Definition:
  – a *set of values* and
  – a *set of operations* that can be applied to those values

• Java is a *strongly-typed* language: compiler & run-time system ensure that programs don’t violate type system rules
  – Compiler catches most illegal operation attempts
  – Java virtual machine catches a few (e.g. invalid casts)

• By contrast, C/C++ not strongly typed
Types in Java

• Primitive: int, double, char, boolean, etc.
• Class
• Interface
• Array
  – Array in itself is distinct type
  – Individual array elements are of some *component* type
• Null
Values in Java

- Every value is one of the following:
  - Value of a primitive type (e.g. 1.4, -9, ‘c’)
  - Reference to a class object
  - Reference to an array
  - null

- Notes:
  - No such thing as a value of an interface type
  - Void is not a data type – just a method tag to indicate no return value
Subtype relationship

• Subtype contains subset of values of a given type
• Can use subtype where supertype is specified
S is a subtype of T if:

1. S and T are the same type
2. S and T are both class types, and T is a direct or indirect superclass of S
3. S is a class type, T is an interface type, and S or one of its superclasses implements T
4. S and T are both interface types, and T is a direct or indirect superinterface of S
5. S and T are both array types, and the component type of S is a subtype of the component type of T
6. S is not a primitive type and T is the type Object
7. S is an array type and T is Cloneable or Serializable
8. S is the null type and T is not a primitive type
Examples

ListIterator is a subtype of Iterator

Container is a subtype of Component

FlowLayout is a subtype of Layout Manager

JButton is a subtype of Component
Examples

• Rectangle[] is a subtype of Shape[]
• int[] is a subtype of Object
• int is not a subtype of long
• long is not a subtype of int
• int[] is not a subtype of Object[]
Array Types

• Recall rule 5: $S[]$ is a subtype of $T[]$ is $S$ is a subtype of $T$

• Example:
  
  Rectangle [] $r = $ new Rectangle[10];
  
  Can store object $r$ in an array of Shape, since
  Rectangle is a subset of Shape:
  
  Shape [] $s = r;$
  
  Variables $r$ and $s$ both refer to the same array
Example continued

What happens if a non-Rectangle Shape is stored in s[]?  
- Throws ArrayStoreException at runtime  
- Every array object remembers its component type  
- Could store a subtype of Rectangle (but not just of Shape) without throwing exception
Primitive type wrappers

• Many services in Java API deal with Objects; for example, ArrayLists are collections of Objects
• Primitive types are not subtypes of Object
• Useful to have classes to “wrap” primitive types so they can avail themselves of Object services
Wrapper classes in Java

• Wrapper classes exist for all primitive types:
  – Integer, Short, Long
  – Byte, Char
  – Float, Double
  – Boolean

• Wrapper classes are immutable
Wrapper class methods

• Constructor take primitive type argument:
  
```java
double num = 3.14159;
Double d = new Double(num);
```

• Can “unwrap” to get primitive value:
  
```java
num = d.doubleValue();
```

• Some wrappers have parse methods that convert from String to primitive:
  
```java
String s = “52406”;
int n = Integer.parseInt(s);
```
Enumerated types

- Type with finite set of values
- C++ has enum mechanism for creating such a type; Java doesn’t
- Can simulate some of the same functionality in Java using typesafe enumeration (example on next slide)
- Only operation on these objects is comparison for equality
Typesafe enumeration example

```java
public class Science {
    private Science (String name) {
        this.name = name;
    }
    private String name;
    public static final Science BIOLOGY =
        new Science("BIOLOGY");
    public static final Science CHEMISTRY =
        new Science("CHEMISTRY");
    public static final Science PHYSICS =
        new Science("PHYSICS");
    ...
}
```
Type inquiry

• Can use instanceof operator to test whether an expression is a reference to an object of a given type or one of its subtypes
• Operator returns true if expression is a subtype of the given type; otherwise (including in the case of a null expression) returns false
Example using instanceof

```java
JPanel picPanel = new JPanel();
ArrayList images = new ArrayList();
...
for (int x=0; x<images.size(); x++)
{
    Object o = images.get(x);
    if (o instanceof Icon)
    {
        Icon pic = (Icon)o;
        picPanel.add(pic);
    }
}
```
Notes on instanceof

- Can test whether value is subtype of given type, but doesn’t distinguish between subtypes
- In previous example, o could be any kind of Icon (since Icon is an interface, couldn’t be an Icon per se)
Getting exact class of object

- Can get exact class of object reference by invoking `getClass()` method
- Method `getClass()` returns an object of type `Class`; can invoke `getName()` on `Class` object to retrieve String containing class name:
  ```java
  System.out.println(o.getClass().getName());
  ```
Class object

- Type descriptor
- Contains information about a given type, e.g. type name and superclass

```
: Employee
  name = "Jane Doe"
  salary = 50000

: Class
  name = "Employee"
  superclass = __null

: Class
  name = "java.lang.Object"
  superclass = null
```
Ways to obtain Class object

• Call `getClass()` on an object reference
• Call static Class method `forName()`:
  ```java
  Class c = Class.forName("java.awt.Rectangle");
  ```
• Add suffix `.class` to a type name to obtain a literal Class object:
  ```java
  Class c = Rectangle.class;
  ```
Notes on Class type

• There is exactly one Class object for every type loaded in the virtual machine
• Class objects describe any type, including primitive types
• Can use == to test whether two Class objects describe same type:
  
  if (o.getClass() == Rectangle.class)
  True if o is exactly a Rectangle (not a subclass)
Arrays and getClass()

• When applied to an array, getClass() returns a Class object that describes the array type
• Use Class method getComponentType() to get a Class object describing the array’s components
• Use Class method isArray() to determine whether or not an object is an array
Arrays and Class.getName()

Array name is built from these rules:

[type

Where type is one of the following:

B: byte  C: char
D: double  F: float
I: int  J: long
S: short  Z: boolean

For non-primitive types, replace “type” with Lname; (where name is the name of the class or interface – note semicolon)