Announcements

• Final Exam: Tues. Dec. 14 @ 3:25pm

Material

• Java type system
• Reflection
Java Type System

• Type System is a set of values and the operations that can be applied to the values

• Strongly typed language: compiler and run-time system check that no operation can execute that violates type system rules

• Compile-time check: call a method that does not exist

  Employee e = new Employee();
  e.clear(); // ERROR

• Run-time check

  e = null;
  e.setSalary(20000); // ERROR
Types in Java

• Primitive types:
  • int  short  long  byte  char  float  double  boolean
• Class types
  • Employee e;
• Interface types
  • ActionListener a;
• Array types
  • String[] s;
• The null type
• Note: void is not a type in Java
Four types of values in Java

1. Value of primitive types
   - 10, 10.5, ‘h’, true
2. Reference to object of a class type
   - new Employee()
3. Reference to array
   - new int[] { 1, 2, 3, 4 }
4. null value is a special type of value
   - null
Primitive Types

• Primitive types are not classes and thus their values cannot be treated like objects.
• Example:
  • Object o = 10;  //illegal

• Java provides wrapper classes
  • Integer for int, etc.

• Use the wrapper to treat primitives like objects
  • Objects o = new Integer(10);  //legal
Type inquiry

- Some languages provide facilities to check types of values at runtime.

- Consider
  - Object o = ...

- What is stored in o? What is the type of the value stored there?
  - Depends on runtime behavior of program

- Java’s runtime system keeps information of type, so it can be checked at runtime
Type safety when casting

- You cast values to convert type
  - Shape e = (Shape) somevalue;

- What if ‘somevalue’ is not of type (or subtype) Shape?
  - Illegal cast exception will occur...

- To test if value can be casted, use `instanceof`
  - if (somevalue instanceof Shape)
• Every object in Java has a type object associated with it at runtime.

• Object e = new Rectangle();
  Class c = e.getClass();
  System.out.println(c.getName());
  // prints java.awt.Rectangle

• Class object describes a type
Class object

- Class object reveals
  - superclass
  - interfaces
  - package
  - names and types of fields
  - names, parameter types, return types of methods
  - parameter types of constructors

- There is a single instance for a given class, never more than one
Getting a Class object

- Three ways to get a Class object
- `getClass()` method gets class of any object, returning an object of type `Class`.
  
  ```java
  Rectangle r = new Rectangle();
  Class c = r.getClass();
  ```

- `Class.forName()` method yields Class object:
  
  ```java
  Class c = Class.forName("java.awt.Rectangle");
  ```

- `.class` suffix yields Class object too
  
  ```java
  Class c = Rectangle.class;
  ```
Some methods

• Some methods in the Class class
  
  Class getSuperclass()
  Class[] getInterfaces()
  Package getPackage()
  Field[] getDeclaredFields()
  Constructor[] getDeclaredConstructors()
  Method[] getDeclaredMethods()

• Note that you can get access to fields, methods, and constructors

• You can also call constructor and methods
Enumerate fields

- Prints the names of all static fields of the Math class

```java
Field[] fields = Math.class.getDeclaredFields();
for (int i = 0; i < fields.length; i++)
    if (Modifier.isStatic(fields[i].getModifiers()))
        System.out.println(fields[i].getName());
```
Enumerate Constructors

- Prints names/parameter types of all Rectangle constructors

```java
for (int i = 0; i < cons.length; i++)
{
    Class[] params = cons[i].getParameterTypes();
    System.out.print("Rectangle(");
    for (int j = 0; j < params.length; j++)
    {
        if (j > 0) System.out.print(",");
        System.out.print(params[j].getName());
    }
    System.out.println(")");
}
```

- Prints

- Rectangle()
- Rectangle(java.awt.Rectangle)
- Rectangle(int, int, int, int)
- Rectangle(int, int)
- Rectangle(java.awt.Point, java.awt.Dimension)
- Rectangle(java.awt.Point)
- Rectangle(java.awt.Dimension)
• Supply method name, array of parameter types (formal arguments)

• Example: Rectangle.contains(int, int)
  Method m = Rectangle.class.getDeclaredMethod(
    "contains",
    new Class[] { int.class, int.class });

• Example: Default constructor
  Constructor c = Rectangle.class.getDeclaredConstructor(
    new Class[] {});
Invoking a method

• Supply implicit parameter, array of explicit parameter values

• Must *wrap* primitive types and unwrap primitive return value

• Example: calling System.out.println

```java
Method m = PrintStream.class.getDeclaredMethod("println",
    new Class[] { String.class } );
m.invoke(System.out,
    new Object[] { "Hello, World!" });
```
• Can obtain object contents at runtime, useful for generic debugging tools, need to gain access to private fields

  Class c = obj.getClass();
  Field f = c.getDeclaredField(name);
  f.setAccessible(true);
  // Throws exception if security manager disallows access
  // Access field value:
  Object value = f.get(obj);
  f.set(obj, value);