2: Everything is an Object

You Manipulate Objects Using References

```java
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`. Read only!
Example

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

Arrays in Java

- Not pointers, like C, but first-class objects
- Checked at run-time for safety
- Covered later

Scoping

```java
{ // <-- Beginning of scope 1
    int x = 12;
    // Only x available
    { // <-- Beginning of scope 2
        int q = 96;
        // Cannot redefine x here!
        // Both x & q available
    } // <-- End of scope 2
    // Only x available
    // q "out of scope"
} // <-- End of scope 1
```
You Never Destroy Objects

- Scope of objects

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

Creating New Data Types: class

- Class keyword defines new data type

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

- Fields

```java
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:
    ```java
    int x = a.f(); // For object a
    ```
The Argument List

- Types of the objects to pass in to the method
- Name (identifier) to use for each one
- Whenever you seem to be passing objects in Java, you're actually passing references

```java
int size(String s) {
    return s.length();
}
```

Naming Conventions

- Part of JavaSoft programming standard
- Words run together, no underscores
- Intermediate words capitalized
- Classes: first letter capitalized
- Methods and variables (including references): first letter lowercase
- Constants: all caps with underscores to separate words (like C).

Building a Java Program

- Before you can build your first Java program you need to understand:
  - Name visibility
  - Using other components
  - The `static` keyword
Name Visibility

- Preventing name clashes: methods and fields are already nested within classes
- What about class names?
- Produce an unambiguous name for each library using your reversed domain name and library path
  - My domain: BruceEckel.com
  - Old: COM.bruceeckel.utility.foibles
  - Later change to all lowercase: com.bruceeckel.utility.foibles

Using Other Components

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

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?

```java
class WithStaticData {
    static int x;
    int y;
    public void doStuff(){
        y = x + y;
    }
}
```

```java
WithStaticData a = new WithStaticData(), b = new WithStaticData(), c = new WithStaticData();
```
**static Methods**

- What if you want a method that can be called for the class, without an object? (“class method”)

```java
class StaticFun {
    static void incr() {
        WithStaticData.x++;
    }
}
StaticFun.incr();
```

**Your First Java Program**

```java
// HelloDate.java
import java.util.*;
public class HelloDate {
    public static void main(String[] args) {
        System.out.println("Hello, it's: ");
        System.out.println(new Date());
    }
}
```

**Comments & Embedded Documentation**

- Javadoc: nice comment-documentation system, extracts tagged comments and turns them into HTML pages
- Used to create Javasoft documentation
- Covered in book