Chapter 2

Constants, Variables, Operators

Types of identifiers

- Reserved words (identifiers reserved for a specific purpose).
- Identifiers that are not reserved words but are already in use.
- Identifiers you make up when writing a program.

Rules

- Use any combination of letters, digits, the underscore ("**_**"), and/or the dollar sign ("$"), but do not start an identifier with a digit.

- Do not embed any **whitespace** in the identifier.

- Do not match any reserved word.

Conventions

- Class names should start with an uppercase letter. Method and parameter names should start with a lowercase letter.

- Use camelcase: an uppercase letter at the beginning of any non-initial word in a multiple-word identifier.

```
getMaximumScore
```

- Use meaningful identifiers.

Reserved words

```
abstract  assert       boolean
char      class        const
else      enum         extends
for       goto         if
interface long         native
protected public       return
switch    synchronized this
try       void         volatile


break      byte    case       catch
continue   default do         double
false      final   finally    float
implements import  instanceof int
new        null    package    private
short      static  strictfp   super
throw      throws  transient  true
while
```

Constants

```
“hello”        String
“20”
“A”

20             int
-7

20.0           double
-1.3
1.2E3  = 1.2 x 10³ = 1200.0

20.0f          float
-1.3f

‘A’            char
```

Arithmetic

```
System.out.println(3 - 2);

System.out.println("3" - "2");

System.out.println("down" + "town");

System.out.println("11" + "7");

System.out.println(11 + "7");
```

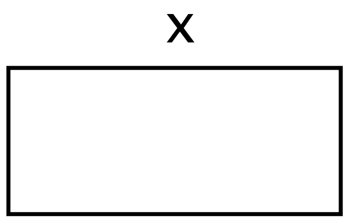
Arithmetic Operators

+	addition
-	subtraction
*	multiplication
/	division
%	remainder (can have integer operands only)

```
System.out.println(2 + 3);

System.out.println(2*3);

System.out.println((2)(3));

System.out.println(2×3);
System.out.println(5/2);

System.out.println(5.0/2.0);

System.out.println(5.0/2);

System.out.println(5%2);

System.out.println("5/2");
System.out.println(6 + 27/3);

System.out.println((6 + 27)/3);

System.out.println(10 - 5 - 3);

System.out.println(10 - (5 - 3));
```

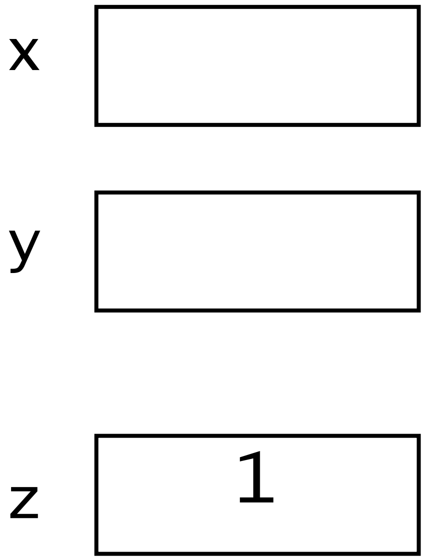
Variables

A **variable** is a named “box” in memory in which a value may be stored.



Declaring variables

```
int x, y, z = 1;
```

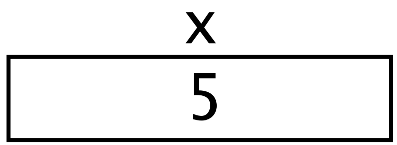


Duplicate declaration

```
int x, y, z;
int x;              // illegal!!!
```

Assignment statement

```
x = 5;

 ...

x = 7;
```



```java
class Variables
{
   public static void main(String[] args)
   {
      int x;
      x = 7;
      System.out.println(x);
      System.out.println(x + 4);
      System.out.println(x);
      System.out.println("x");
      x = 20;
      System.out.println(x);
      x = x + 1;
      System.out.println(x);
      x++;
      System.out.println("x = " + x);
      x--;
      System.out.println("x = " + x);

      double y, z;
      y = 5.0;
      y = y/2.0;
      System.out.println(y);
      z = y/2.0;
      System.out.println(z);
   }
}
```

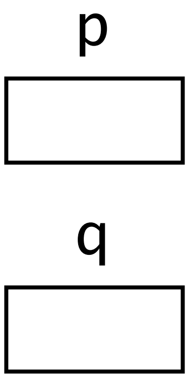
Illegal

```
class Bad
{
   public static void main(String[] args)
   {
      m = 2;     // using before declaring
      int m;

      int n, m;
      m = n + 1; // n has no value
   }
}
```

Creating an object

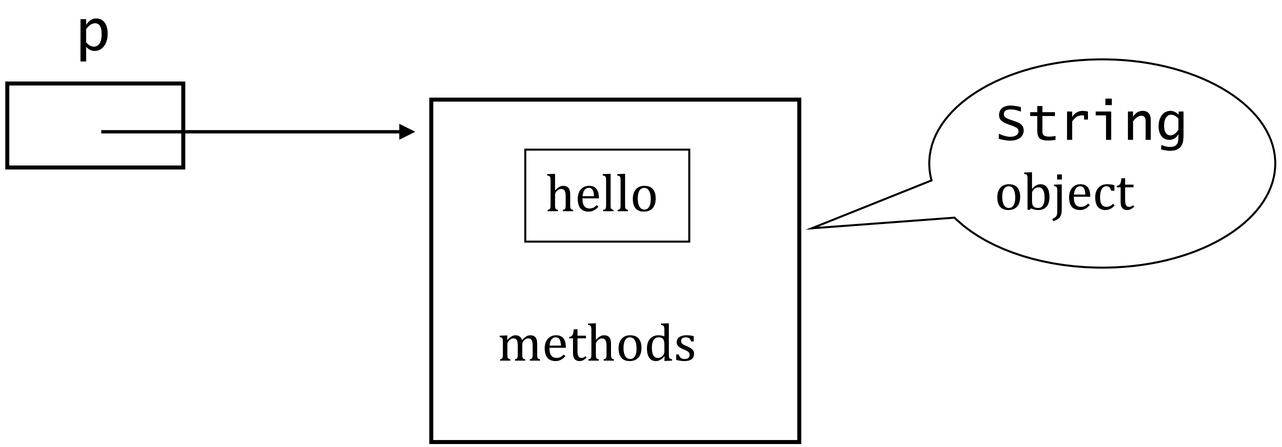
First create a **reference variables**:

```
String p, q;
```



Now create the object

```
p = new String(“hello”);
```

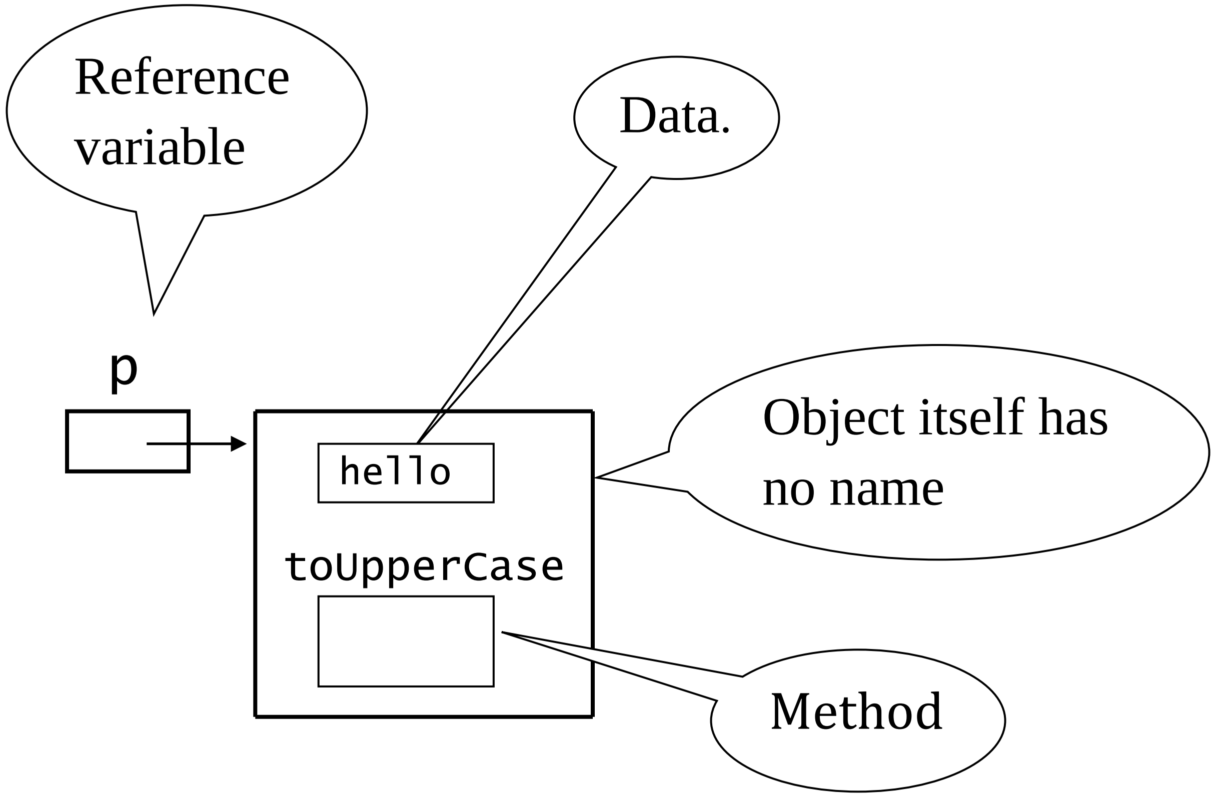


An object **encapsulates** data and methods that operate on that data.

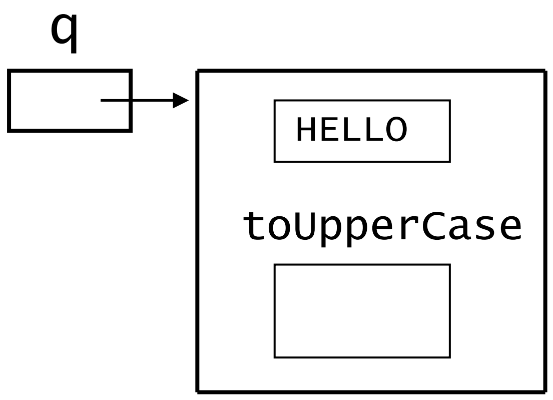
Complete program

```
class Program2
{
   public static void main(String[] args)
   {
      String p, q;
      p = new String("hello");
      q = p.toUpperCase();
      System.out.println(p);
      System.out.println(q);
      String r = new String("bye");
      String s = "all done";
      System.out.println(r);
      System.out.println(s);
   }
}
```

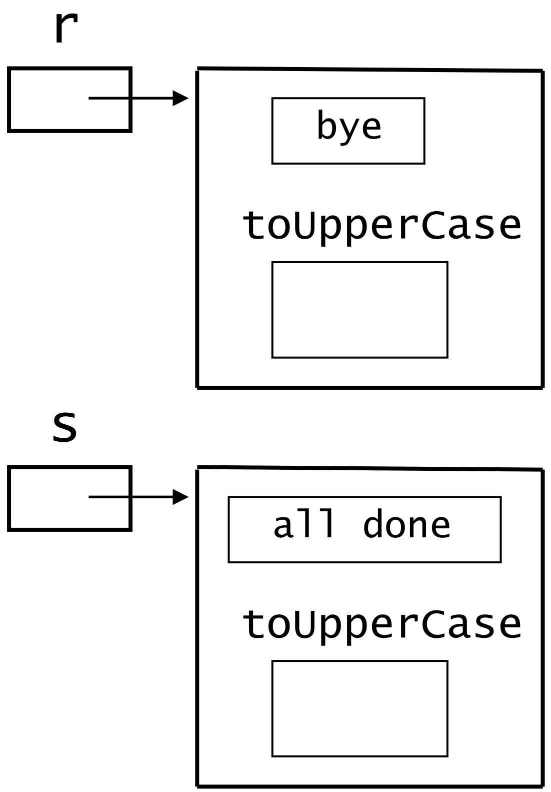
```
p = new String("hello");
```



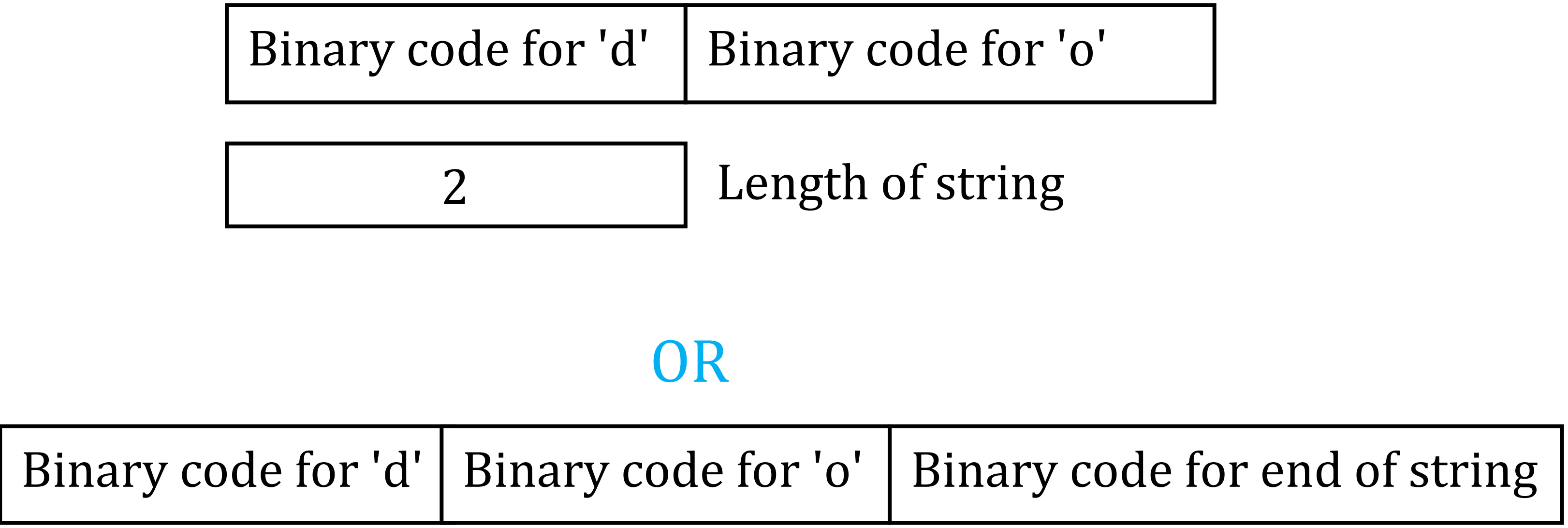
```
q = p.toUpperCase();
```



```
String r = new String("bye");
String s = "all done";
```



Representing "do" in memory



Information hiding: We don't have to know representation to use `String` class.