Building a class

A very basic class describing a student:

```
class Student {
  String name;
  String stunum;
  int numCourses;
  int numPassed;
  double avgMark;
}
public class TestStu {
  public static void main (String[] args) {
    Student stu1 = new Student();
    Student stu2;
    stu2 = new Student();
    stu1.name = "Jim";
    stu2.name = "Mary";
  }
}
```

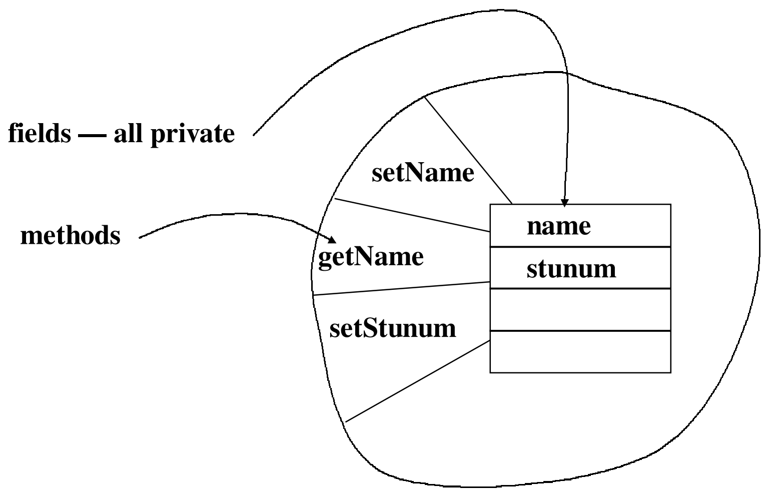
Protecting information

```
class Student {
  private String name;
  private String stunum; ... // ALL fields private
  ...
  // setName: a "set method" or "mutator"
  public void setName (String newName) {
    name = newName;
  }
  // getName: a "get method" or "accessor"
  public String getName () }
    return name;
  }
}
```

And now, in main( ):

```
  stu1.setName("Jim");
  stu2.setName("Mary");
```

A picture



Constructors

```
Student stu1 = new Student(); // 1
stu1.setName("Jim");          // 2
```

Between stmt 1 and stmt 2, stu1 has no name!

(Why does this matter?)

The fix:

```
Student stu1 = new Student("Jim", "99001");
```

Here's the ***constructor*** (which also sets the student number):

```
public Student (String theName, String theNum) {
  name = theName;
  stunum = theNum;
}
```

- Can we still say, **`stuX = new Student();`?**
- What's the return type of a constructor?

The toString( ) method

A useful standard method, present in all classes:

```
public String toString() {...}
```

For example, in the Student class:

```
public String toString() {
  String result = "Information about " + name;
  result += " (" + stunum + "):"
  result += "\n courses taken " + numCourses;
  if (numCourses > 0)
    result += ", average = " + avgMark;
  return result;
}
```

For example, now `System.out.println(stu1.toString());` might print:

```
   Information about Jim (99001):
    courses taken = 4, average = 73
```

Using toString( ) by default

Instead of `System.out.println(stu1.toString());` you can just say

```
System.out.println(stu1);
```

Changing an implementation

We've stored a Student's name as a single string, with a getName method to retrieve the value.

What if we want to let the caller get just the family name?

- We need two public methods:
  ```
  public String getName()
  public String getFamilyName()
  ```
- We have our choice of private fields:
  ```
  private String name;
  ```
  or
  ```
  private String familyName;
  private String givenNames;
  ```

Which should we use?

Either one might be right!

getFamilyName( ) with one "name" field

We will assume these specifications for the "name" field:
The name field consists of these parts, in this order:
1) the family name, possibly including some (single) blank characters
2) a double blank " " if necessary to separate the family name from the given names.
3) zero or more given names
4) maybe some trailing blanks

At least part (1) is required, but the field may omit later parts.
(Or: "Parts 1 to N are required, where N >= 1. If part k is omitted, all parts j must be omitted, with j >= k. Or …?)

Examples:

name:	**`CLARKE  JIM`**	family name:	**`CLARKE`**
	`CLA R  JIM`		**`CLA R`**
	`CLA`		**`CLA`**
	`CLAb`		**`CLA`**
	`bb`		**`""`** (empty string)

Keep it simple on the first try

```
int pos = 0; // the current character position
String result = ""; // the family name
   // that we're building up

while (pos < name.length()) {
  if (name.charAt(pos) == ' ') {
    if (name.charAt(pos+1) == ' ') // not "="
      return result;
  }
  else
    result += name.charAt(pos);
  pos++;  // pos = pos + 1;
}
```

- What about "CLAbR"?
- What about "CLA"?
- What about "CLAb"?