Computer Science 202
Introduction to Programming
The College of Saint Rose
Fall 2013

Topic Notes: The `switch` Statement

We have seen that a common pattern in programming is to have a series of statements of the form:

```
if (x == 0) {
  // do stuff for x == 0
}
else if (x == 1) {
  // do stuff for x == 1
}
else if (x == 2) {
  // do stuff for x == 2
}
...
else if (x == 8) {
  // do stuff for x == 8
}
else {
  // do stuff when x is none of the above
}
```

Let's look at an example where this occurs. Consider a program that tells you which Computer Science faculty member you can find in each of the offices in the Albertus 400 suite.

See Example: CSOfficesIfElse

Java (and many other languages) provide a special construct we can use in situations like this that can be a bit more convenient.

```
switch (x) {
  case 0:
    // do stuff for x == 0
    break;
  case 1:
    // do stuff for x == 1
    break;
  case 2:
    // do stuff for x == 2
    break;
```

```
...
  case 8:
    // do stuff for x == 8
    break;
  default:
    // do stuff when x is none of the above
    break;
}
```

This works only when the comparison if for equality and we are using one of these data types: `char`, `byte`, `short`, or `int`. So far, we have only used `int` variables from among this group. Note that it does not work for `double` or `String` values.

Also note that each `case` is ended by a special statement: `break;`

If we rewrite the example to use a `switch` statement, it would look like this:

See Example: CSOfficesSwitch

If we mistakenly leave out a `break;` statement, Java will "fall through" to the next `case`. Sometimes this is handy and just what we want, but the vast majority of the time, we want a `break;` at the end of case `case`.

One situation where this does come in handy is when we want to do the same thing for multiple cases:

See Example: LittlePrimes