Type Casting

```
int j;
double e;
...
j = (int)e; // OK
```

- This is fine, *if* the double can “fit” into an integer
- Otherwise, the compiler does its “best” to make it fit!

Type Casting (cont.)

```
int j, k;
double d = 3.9,
       e = 2E20;
j = (int)d; // OK
k = (int)e; // Too big
System.out.println("j = " + j);
System.out.println("k = " + k);
```

```
j = 3
k = 2147483647
```

Assignments

- Unusually, assignment in Java is an *expression*, not a statement
- So,

```
a = 3
```

- has a value!

```
int i, j, k;
i = ( j = (k = 0));
i = j = k = 0; // Brackets not needed
```

Incrementing and Decrementing

- Common operations:

```
a = a + 1;  b = b - 1;
```

- Shorthand:

```
a++;  b--;
```

Incrementing and Decrementing (cont.)

- Can also be used in expressions:

```
int i = 5, j, k;
i++;            // i is now 6
j = i++;        // i is 7, j is 6
k = --j;        // j is 5, k is 5
```

Chapter 3
Flow of Control

Preliminaries

- Look at syntax (not concepts)
- Control structures allow for *single* statements
  - Braces allow us to use *compound statements*
  - Usually use braces always

```
if (x == 3)
  { System.out.print(”x is 3”);
  }
```

The `if` Statement

```
if (boolean_expression)
   { statement; ...
   }
else
   { statement; ...
   }
```

- The brackets around the ***`boolean_expression`*** are required
- The **`else`** part is optional

The **`switch`** Statement

```
switch (expression)
  { case label: ... case label:
      statements;   break;
    case label: …
      statements;   break;
    case label: ...
      statements;   break;
    default:
      statements;   break;
  }
```

- *Must* have the **`break`** statements

An Example of a **`switch`** Statement

```
switch (num)
  { case 0:
      System.out.print("Zero");
      break;
    case 1: case 3: case 5: case 7: case 9:
      System.out.print("Odd");
      break;
    case 2: case 4: case 6: case 8:
      System.out.print("Even");
      break;
    default:
      System.out.print("Not 0-9");
      break;
  }
```

The `while` Statement

```
while (boolean_expression)
   { statement; ...
   }
```

- Pretest loop

The **do** Statement

```
do
   { statement; ...
   } while (boolean_expression)
```

- Post-test loop
- Executes at least once

The `for` Statement

```
for (expr1; expr2; expr3)
   { statement; ...
   }
```

- Three controlling expressions, separated by semicolons
  - *initialisation:* executed once, before the loop begins
  - *test:* executed before each iteration; *must* be a boolean expression
  - *increment:* executed at the end of each loop

The `for` Statement (cont.)

- Any of the controlling expressions may be left out
  - If the test is omitted it is taken to be permanently true

The **`break`** Statement

- Essential in **`switch`** statements
- Also used in loops (**`for`**, **`while`** and **`do`**)
  - Early exit
- E.g.

```
while (...)
  { ...
    if (...)
      break; // Jump out
    ...
  } // while
```

The **continue** Statement

- Used in loops (**for**, **while** and **do**)
- Restarts a loop, checking the condition

```
for (int k = 0; k < 100; k=k+1)
  { int r = (int)(Math.random() * 100);
    if (r % 3 == 0) // r is a multiple of 3
      { num = num + 1;
        continue; // Restart
      }
    sum  = sum + r;
  } // for
```