Properties of conditionals

- In the following, `C` is any boolean expression, `P`, `Q`, `R` , `S`, and `T` are any list of statements.

```
P;
if (C) {
  Q;
  R;
}
else{
  Q;
  S;
}
T;
```

Properties of conditionals

is equivalent to

```
P;
Q;
if (C) {
  R;
}
else {
  S;
}
T;
```

if and only if the statements in `Q` do not modify the variables in `C`

Properties of conditionals

- Consider the following:

```
boolean high = false;
double altitude;
altitude = Keyboard.readDouble();
System.out.println("Begin");
if (altitude > 2000.0) {
  high = true;
  System.out.println("It is high");
}
else {
  high = true;
  System.out.println("It is low");
}
```

Properties of conditionals

- It is equivalent to:

```
boolean high = false;
double altitude;
altitude = Keyboard.readDouble();
System.out.println("Begin");
high = true;
if (altitude > 2000.0) {
  System.out.println("It is high");
}
else {
  System.out.println("It is low");
}
```

Properties of conditionals

- Consider the following:

```
double altitude;
altitude = Keyboard.readDouble();
System.out.println("Begin");
if (altitude > 2000.0) {
  altitude = altitude - 500.0;
  System.out.println("It is high");
}
else {
  altitude = altitude - 500.0;
  System.out.println("It is low");
}
```

Properties of conditionals

- It is *not* equivalent to:

```
double altitude;
altitude = Keyboard.readDouble();
System.out.println("Begin");
altitude = altitude - 500.0;
if (altitude > 2000.0) {
  System.out.println("It is high");
}
else {
  System.out.println("It is low");
}
```

Conditionals

```
int x;
boolean b;
//...
if (b) {
  x=3;
}
else {
  x=4;
}
```

is equivalent to

```
x=4;
if (b) {
  x=3;
}
```

Conditionals

```
int x,y;
boolean b;
//...
if (b) {
  x=3;
}
else {
  y=4;
}
```

is *not* equivalent to

```
y=4;
if (b) {
  x=3;
}
```

Constants

- To enforce that a variable cannot change we declare it as a constant:

```
final type variable = expression;
```

- The variable must be initialised

```
final double PI = 3.1415;
PI = 2 * PI; // Error
```

- A variable declared as final is a constant and cannot ocurr on the left-hand side of an assignment statement

- It is common practice (but not mandatory) to name constants in all capitalized letters.

Abstraction

- Abstraction:

  "disassociated from any specific instance" - Webster's dictionary

- To abstract is to make something independent of particular cases

- Variables give us a basic mechanism for abstraction:
  - A concrete definition:

$$tax = 3217.50 + 0.28 \times (income - 21450.0)$$

  - An abstract definition:

$$tax = base + rate \times (income - cutoff)$$

- In software, abstraction facilitates reusability and makes it easier to maintain.

The random method

- The method

  ```
  static double random()
  ```

  from the Math class returns a random number between 0 and 1 (including 0 but excluding 1)

- It can be used for giving random integers in any interval by means of casting

  ```
  int coin;
  coin = (int)(Math.random() * 2);

  int die;
  die = (int)(Math.random() * 6 + 1);
  ```

Large conditionals

```
int die;
die = (int)(6 * Math.random() + 1);

if (die == 1)
  System.out.println(“Excellent”);
else
  if (die == 2)
    System.out.println(“Good”);
  else
    if (die == 3)
      System.out.println(“OK”);
    else
      if (die == 4)
        System.out.println(“Ah...”);
      else
        if (die == 5)
          System.out.println(“Bad”);
        else
          if (die == 6)
            System.out.println(“Terrible”);
```

Large conditionals

```
int die;
die = (int)(6 * Math.random() + 1);

if (die == 1)
  System.out.println(“Excellent”);
else if (die == 2)
  System.out.println(“Good”);
else if (die == 3)
  System.out.println(“OK”);
else if (die == 4)
  System.out.println(“Ah...”);
else if (die == 5)
  System.out.println(“Bad”);
else if (die == 6)
  System.out.println(“Terrible”);
```

The switch statement

```
int die;
die = (int)(6 * Math.random() + 1);
switch (die) {
  case 1:
    System.out.println(“Excellent”);
    break;
  case 2:
    System.out.println(“Good”);
    break;
  case 3:
    System.out.println(“OK”);
    break;
  case 4:
    System.out.println(“Ah...”);
    break;
  case 5:
    System.out.println(“Bad”);
    break;
  case 6:
    System.out.println(“Terrible”);
    break;
}
```

Large conditionals

```
int die;
die = (int)(6 * Math.random() + 1);

if (die == 1)
  System.out.println(“Excellent”);
else if (die == 2)
  System.out.println(“Good”);
else if (die == 3)
  System.out.println(“OK”);
else if (die == 4)
  System.out.println(“Ah...”);
else if (die == 5)
  System.out.println(“Bad”);
else
  System.out.println(“Terrible”);
```

The switch statement

```
int die;
die = (int)(6 * Math.random() + 1);
switch (die) {
  case 1:
    System.out.println(“Excellent”);
    break;
  case 2:
    System.out.println(“Good”);
    break;
  case 3:
    System.out.println(“OK”);
    break;
  case 4:
    System.out.println(“Ah...”);
    break;
  case 5:
    System.out.println(“Bad”);
    break;
  default:
    System.out.println(“Terrible”);
    break;
}
```

The switch statement

- Just another form of conditional

```
switch (integer_or_character_expression) {
  case integer_or_character_expression_1:
    list_of_statements_1;
    break;
  case integer_or_character_expression_2:
    list_of_statements_2;
    break;
  case integer_or_character_expression_3:
    list_of_statements_3;
    break;
   ...
  default:
    list_of_statements_n;
}
```

The switch statement

- Semantics:

1. Evaluate the condition,
   (a) compare it with each case
   (b) if a case matches, the corresponding list of statements is executed
      i. if there is a break statement, the switch stops and computation continues directly after the switch.
      ii. if there is no break statement in the list, execution continues with the next case

The switch statement

```
int die;
die = (int)(6 * Math.random() + 1);
switch (die) {
  case 1:
    System.out.println(“Excellent”);
    break;
  case 2:
    System.out.println(“Good”);
  case 3:
    System.out.println(“OK”);
  case 4:
    System.out.println(“Ah...”);
    break;
  case 5:
    System.out.println(“Bad”);
    break;
  default:
    System.out.println(“Terrible”);
    break;
}
```

The switch statement

- If the break statement is included,

```
switch (C) {
  case E1:
    S1;
    break;
  case E2;
    S2;
    break;
  case E3;
    S3;
    break;
  ...
  default:
    Sn;
}
```

  is equivalent to

The switch statement

```
if (C == E1) S1;
else if (C == E2) S2;
else if (C == E3) S3;
...
else Sn;
```

Switch conditions

- An integer expression is an arithmetic expression of type int, short, long or byte, e.g.

```
3
5+3*-2
x*(7/2)          // (if x is of type int)
(int)'A'
s.length()       // (if s is a String)
```

- The expression `(int)'A'` has as value the ASCII or Unicode number for the character `'A'`

Character expressions

- A character expression is an expression of type char

```
'a'
'B'
'8'
' '
'd' + 2
(char)65
s.charAt(3)              // if s is a String
```

- The expression `'d' + 2` has as value the character `'f'`
- The expression `(char)65` has as value the character corresponding to the ASCII or Unicode number 65 (`'a'`)
- Character expressions can be used in relational expressions (Their ASCII or Unicode value is compared):

```
'm' <= 'p'
'D' > 'A'
'a' < 'A'
```

Character expressions

```
String sentence;
char c;
boolean letter = false, digit = false;

sentence = Keyboard.readString();
c = sentence.charAt( sentence.length() - 1 );

if (‘A’ <= c && c <= ‘Z’ || ‘a’ <= c && c <= ‘z’)
  letter = true;
else if (‘0’ <= c && c <= ‘9’)
  digit = true;
```

Character expressions

```
String sentence;
char c;

sentence = Keyboard.readString();
c = sentence.charAt( sentence.length() - 1 );

if (‘A’ <= c && c <= ‘Z’) {
  c = (char)(c + (‘a’ - ‘A’));
  // c is a lower case letter
}
```

Character expressions

```
String sentence;
char c;

sentence = Keyboard.readString();
c = sentence.charAt( sentence.length() - 1 );

if (‘a’ <= c && c <= ‘z’) {
  c = (char)(c + (‘A’ - ‘a’));
  // c is an upper case letter
}
```

Switch conditions

```
String name;
name = Keyboard.readString();

switch( name.charAt(3) - 2 ) {
  case ‘e’:
    System.out.println(“Helloooo”);
    break;
  case ‘h’:
    System.out.println(“Noooo”);
    break;
  case ‘z’:
    System.out.println(“OK”);
}
```

Character expressions

```
String sentence;
char c;
boolean vowel;

sentence = Keyboard.readString();
sentence = sentence.toLowerCase();
c = sentence.charAt( sentence.length() - 1 );

switch (c) {
  case ‘a’:
  case ‘e’:
  case ‘i’:
  case ‘o’:
  case ‘u’:
    vowel = true;
    break;
  default:
    vowel = false;
}
```

Statements

- Variable declaration

```
type variable;
```

- Assignment

```
variable = expression;
```

- Method invocation

```
objectreference.methodname(parameters);
```

or

```
classname.methodname(parameters);
```

- Conditional

```
if (condition) block;
```

or

```
if (condition) block1; else block2;
```

- Loop

Loops

- The loop is a statement used to describe a task which is *repetitive*
- For example: print the first 100 odd integers

```
System.out.println(1);
System.out.println(3);
System.out.println(5);
System.out.println(7);
System.out.println(9);
System.out.println(11);
System.out.println(13);
//...
```

- What if we want to print the first 1000 odd numbers?
- What if the user is supposed to give the program the number of odd numbers?

Loops

- The basic loop statement:

```
while (boolean_expression) {
  list_of_statements;
}
```

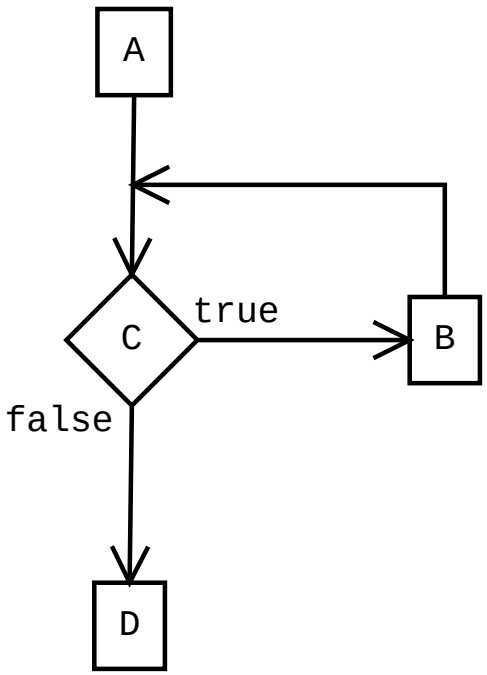
- Semantics: the execution of a while loop proceeds as follows:

1. The boolean expression is evaluated
   (a) If it is false,
      i. the loop stops
      ii. and computation proceeds directly after the loop
   (b) If it is true,
      i. the list of statements is executed,
      ii. and when finished, the whole process is repeated from step 1

Loops

```
A;
while (C) {
  B;
}
D
```

- Control flow diagram:



Loops

```
int counter, number;
counter = 1;
number = 1;
while (counter <= 100) {
    System.out.println(number);
    number = number + 2;
    counter++;
}
System.out.println(“Done”);
```

Loops

```
int counter, number;
counter = 1;
number = 1;
while (counter <= 3) {
    System.out.println(number);
    number = number + 2;
    counter++;
}
System.out.println("Done");
```

counter	number
-	-

(This table shows the values of the variables just before the statement in red is executed)

Printed:

Loops

```
int counter, number;
counter = 1;
number = 1;
while (counter <= 3) {
    System.out.println(number);
    number = number + 2;
    counter++;
}
System.out.println("Done");
```

counter	number
1	-

Printed:

Loops

```
int counter, number;
counter = 1;
number = 1;
while (counter <= 3) {
    System.out.println(number);
    number = number + 2;
    counter++;
}
System.out.println("Done");
```

counter	number
1	1

Printed:

Loops

```
int counter, number;
counter = 1;
number = 1;
while (counter <= 3) {
    System.out.println(number);
    number = number + 2;
    counter++;
}
System.out.println("Done");
```

counter	number
1	1

Printed:

Loops

```
int counter, number;
counter = 1;
number = 1;
while (counter <= 3) {
    System.out.println(number);
    number = number + 2;
    counter++;
}
System.out.println("Done");
```

counter	number
1	1

Printed:

1

Loops

```
int counter, number;
counter = 1;
number = 1;
while (counter <= 3) {
    System.out.println(number);
    number = number + 2;
    counter++;
}
System.out.println("Done");
```

counter	number
1	3

Printed:

1

Loops

```
int counter, number;
counter = 1;
number = 1;
while (counter <= 3) {
    System.out.println(number);
    number = number + 2;
    counter++;
}
System.out.println("Done");
```

counter	number
2	3

Printed:

1

Loops

```
int counter, number;
counter = 1;
number = 1;
while (counter <= 3) {
    System.out.println(number);
    number = number + 2;
    counter++;
}
System.out.println("Done");
```

counter	number
2	3

Printed:

1

Loops

```
int counter, number;
counter = 1;
number = 1;
while (counter <= 3) {
    System.out.println(number);
    number = number + 2;
    counter++;
}
System.out.println("Done");
```

counter	number
2	3

Printed:

1
3

Loops

```
int counter, number;
counter = 1;
number = 1;
while (counter <= 3) {
    System.out.println(number);
    number = number + 2;
    counter++;
}
System.out.println("Done");
```

counter	number
2	5

Printed:

1
3

Loops

```
int counter, number;
counter = 1;
number = 1;
while (counter <= 3) {
    System.out.println(number);
    number = number + 2;
    counter++;
}
System.out.println("Done");
```

counter	number
3	5

Printed:

1
3

Loops

```
int counter, number;
counter = 1;
number = 1;
while (counter <= 3) {
    System.out.println(number);
    number = number + 2;
    counter++;
}
System.out.println("Done");
```

counter	number
3	5

Printed:

1
3

Loops

```
int counter, number;
counter = 1;
number = 1;
while (counter <= 3) {
    System.out.println(number);
    number = number + 2;
    counter++;
}
System.out.println("Done");
```

counter	number
3	5

Printed:

```
1
3
5
```

Loops

```
int counter, number;
counter = 1;
number = 1;
while (counter <= 3) {
    System.out.println(number);
    number = number + 2;
    counter++;
}
System.out.println("Done");
```

counter	number
3	7

Printed:

1
3
5

Loops

```
int counter, number;
counter = 1;
number = 1;
while (counter <= 3) {
    System.out.println(number);
    number = number + 2;
    counter++;
}
System.out.println("Done");
```

counter	number
4	7

Printed:

1
3
5

Loops

```
int counter, number;
counter = 1;
number = 1;
while (counter <= 3) {
    System.out.println(number);
    number = number + 2;
    counter++;
}
System.out.println(“Done”);
```

counter	number
4	7

Printed:

```
1
3
5
Done
```

Loops

```
int counter = 1;
int number = 1;
while (counter <= 10000) {
    System.out.println(number);
    number = number + 2;
    counter++;
}
System.out.println(“Done”);
```

Loops

```
int maximum = Keyboard.readInt();
int counter = 1;
int number = 1;
while (counter <= maximum) {
    System.out.println(number);
    number = number + 2;
    counter++;
}
```

Loops

- A loop may not terminate

```
int maximum = Keyboard.readInt();
int counter = 1;
int number = 1;
while (counter <= maximum) {
    System.out.println(number);
    number = number + 2;
}
```

- A loop will not terminate if its condition is always true

The end