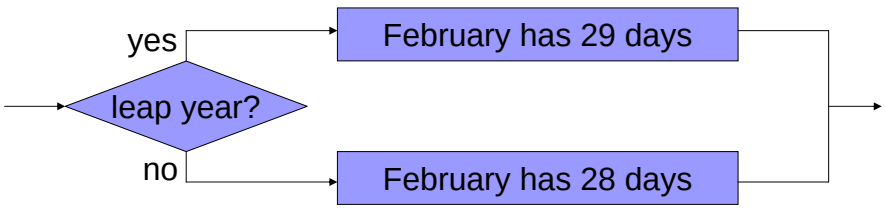
Selection

Hussein Suleman
UCT Dept of Computer Science
CS115 ~ 2003

What is Selection?

UCT-CS

- Making choices in the flow of execution of a program
  - e.g., if it is a leap year then there are 29 days in February – otherwise there are 28



Conditional expressions

UCT-CS

- Selections are made on the basis of expressions that must evaluate to true or false (boolean)
- Relational operators always return boolean values, e.g.:
  - answer > 1.0
  - numberOfPeople <= 14
  - month == 12 // note: this is not the same as "="
  - date != 13 // not equal
  - money >= 5000

The "if" statement

UCT-CS

```
if (boolean_expression)
{
   statements …
}
else
{
   statements …
}
```

Example usage

UCT-CS

```
if (month == 12)
{
   System.out.println (“Hoorah! No classes”);
}
else
{
   System.out.println (“:-(”);
}
```

Another example

UCT-CS

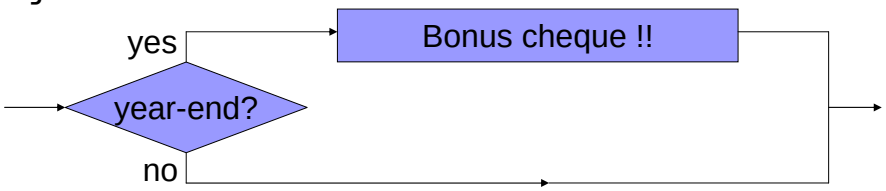
```
if (year < 2000)
{
   fearFactor = 1;
}
else
{
   fearFactor = 0;
}
if (fearFactor == 1)
{
   System.out.println (“be afraid – be very afraid”);
}
else
{
   System.out.println (“it’s OK! no Y2K bug!”);
}
```

Shortcuts I

UCT-CS

- No else part

```
if (numberOfStudents > 150)
{
   System.out.println (“Full!”);
}
```



Shortcuts II

UCT-CS

- Only one statement in block – can leave out the braces

```
if (numberOfStudents > 150)
   System.out.println (“Full!”);
else
   System.out.println (“Not full”);
```

More Data Types

UCT-CS

- char – stores a single character
  - char literals are enclosed in single quotes
  - e.g., `char aLetter = 'a';`
- boolean – stores only *true* or *false* values
  - e.g., `boolean iLikeCSC115 = true;`

```
if (iLikeCSC115)
{
   iEatWeetbix = true;
}
```

Issues with Strings

UCT-CS

- You cannot compare two strings like other types of data
  - i.e., `"Hello" == "Hello"` may not work !
- Instead, use methods in String class
  - `"Hello".compareTo("Hello") == 0`
  - `"Hello".equals ("Hello")`
  - `aString.compareTo ("somevalue") == 0`
  - `aString.equals ("somevalue")`

Nested “if” statement

UCT-CS

```
String password = Keyboard.readString();
if (password.equals (realPassword))
{
   if (name.equals (“admin”))
   {
      loggedIn = superPrivileges = true;
   }
}
else
{
   System.out.println (“Error”);
}
```

Dangling Else

UCT-CS

- Compiler cannot determine which “if” an “else” belongs to if there are no braces

```
String password = Keyboard.readString();
if (password.equals (realPassword))
   if (name.equals (“admin”))
      loggedIn = superPrivileges = true;
else
   System.out.println (“Error”);
```

- Java matches else with *last unfinished if*
- Moral: Use shortcuts at your own risk – or don’t !

Multiway selection

UCT-CS

- Multiple conditions, each of which causes a different block of statements to execute
- Can be used where there are more than 2 options

```
if (condition1)
{
   statements …
}
else
{
   if (condition2)
   {
      statements …
   }
   else
   …
}
```

“if” ladder

UCT-CS

- Just a nicer way to write multiway selection

```
if (operation == ‘a’)
{
   answer = first + second;
}
else if (operation == ‘s’)
{
   answer = first – second;
}
else if (operation == ‘m’)
{
   answer = first * second;
}
```

The “switch” statement

UCT-CS

- Selects among different statements based on a single integer or character expression
- Each set of statements starts in “case” and ends in “break” because switch does not use {}s
  - break passes control to statement immediately after switch
- “default” applies if none of the cases match

Sample switch statement

UCT-CS

```
switch (SouperSandwichOrder)
{
   case 1 : cheese = 1;
            break;
   case 2 : cheese = 1;
            tomato = 1;
            break;
   case 3 : cheese = 1;
            tomato = 1;
            chukka = 1;
            break;
   default : cheese = 1;
            break;
}
```

“break” optimisation

UCT-CS

- If break is omitted, control continues to next statement in the switch

```
switch (SouperSandwichOrder)
{
   case 3 : chukka = 1;
   case 2 : tomato = 1;
   case 1 :
   default : cheese = 1;
}
```

Characters in “switch”

UCT-CS

```
char Operation = Keyboard.readChar (“What to do?”);
switch (Operation)
{
   case ‘a’ : answer = a + b;
              break;
   case ‘s’ : answer = a – b;
              break;
   case ‘m’ : answer = a * b;
              break;
   case ‘d’ : if (b != 0)
              {
                 answer = a / b;
                 break;
              }
   default :  answer = 0;
              System.out.println (“Error”);
              break;
}
```

Boolean operators

UCT-CS

Boolean Algebra	Java	Meaning
AND	&&	true if both parameters are true
OR	||	true if at least one parameter is true
NOT	!	true if parameter is false; false if parameter is true;

Operator precedence

UCT-CS

- Now that we have seen how operators can be mixed, we need precedence rules for all operators
  - () (highest precedence – performed first)
  - !
  - * / %
  - + -
  - < <= > >=
  - == !=
  - &&
  - ||
  - = (lowest precedence – performed last)

Reversing expressions

UCT-CS

- Use ! operator to reverse meaning of boolean expression, e.g.,

```
if (mark >= 0)
{
   // do nothing
}
else
   System.out.println (“Error”);
```

- Instead, invert the condition

```
if (! (mark >= 0))
   System.out.println (“Error”);
```

- Can we do better ?

Boolean operator example

UCT-CS

```
boolean inClassroom, isRaining;
…
if (inClassroom && isRaining)
   System.out.println (“Lucky!”);
…
if (! inClassroom && isRaining)
   System.out.println (“Wet and miserable!”);
…
if (! isRaining && ! inClassroom)
   System.out.println (“Happy!”);
```

Boolean expression example

UCT-CS

```
int marks;
char symbol;
…
if (marks >= 75)
   symbol = ‘A’;
…
if (marks >= 65 && marks <75)
   symbol = ‘B’;
…
if (marks < 0 || marks > 100)
{
   symbol = ‘X’;
   System.out.println (“Invalid mark!”);
}
```

DeMorgan’s Laws

UCT-CS

- !(A && B) = !A || !B
- !(A || B) = !A && !B
- Invert the whole expression, the operators and the operands
  - !(A … B) → (A … B)
  - A → !A
  - && → ||
- Use this tranformation to simplify expressions by removing “!”s wherever possible

Simplification

UCT-CS

- Apply DeMorgan's Laws to simplify

```
(! (mark >= 0 && mark <= 100))
(! (mark >= 0)) || (! (mark <= 100))
(mark < 0 || mark > 100)
```

- Apply DeMorgan's Laws to simplify

```
! ( salary < 10000 || ! me.bigChief ())
(! (salary < 10000)) && (!! me.bigChief ())
salary >= 10000 && me.bigChief ()
```