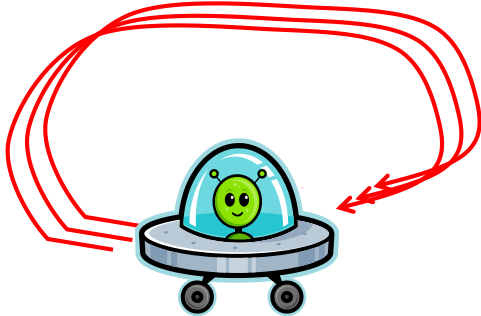


Other conditional and loop constructs



Fundamentals of Computer Science • Keith Vertanen • Copyright © 2013

Overview

- Current loop constructs:
 - for, while, do-while
- New loop constructs
 - Get out of loop early: **break**
 - Skip rest of current loop: **continue**
- Current conditionals:
 - if, if-else, if-else if
- New conditional:
 - When you have many choices: **switch**

Breaking out

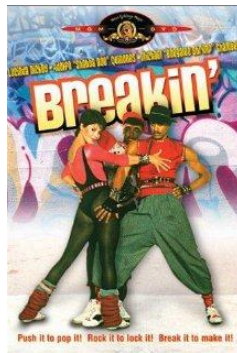
- Loops normally go until loop condition false

```
int i = 0
while (i < 100)
{
    // Do some stuff
    i++;
}

for (int i = 0; i < 100; i++)
{
    // Do some stuff
}
```

- **break statement**

- Exit a loop immediately
- No iteration, no increment
- No condition check
- Straight to the code after loop



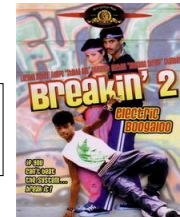
3

Breaking out 2

- **break statement**
 - Terminates enclosing loop: for, while, or do-while
 - Goal: **sum data array, check for invalid negative values**

```
int [] data = {1, 2, 10, 5, -1, 5, 0};
int i = 0;
int sum = 0;
while (???1???)
{
    if (???2???)
        break;
    sum += data[i];
    i++;
}
if (???3???)
    System.out.println("Invalid data!");
else
    System.out.println("Sum: " + sum);
```

This lists tells Java how big the array should be and the initial values in each element.

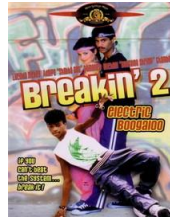


4

Breaking out 2

- **break statement**
 - Terminates enclosing for-loop, while-loop, do-while loop
 - Goal: **sum data array, check for invalid negative values**

```
int [] data = {1, 2, 10, 5, -1, 5, 0};
int i = 0;
int sum = 0;
while (i < data.length)
{
    if (data[i] < 0)
        break;
    sum += data[i];
    i++;
}
if (i < data.length)
    System.out.println("Invalid data!");
else
    System.out.println("Sum: " + sum);
```



5

Skipping and continuing loop

- **continue statement**
 - Skip rest of for-loop, while-loop, do-while body
 - Goal: **sum data array, skipping invalid negative values**

```
int [] data = {1, 2, 10, 5, -1, 5, 0};
int i = 0;
int sum = 0;
for (i = 0; i < data.length; i++)
{
    if (data[i] < 0)
        ???1???;
    sum += data[i];
}
System.out.println("Sum: " + sum);
```

What
statement goes
at ???1???

What does it
print?

6

Skipping and continuing loop

- **continue statement**
 - Skip rest of for-loop, while-loop, do-while body
 - Goal: **sum data array, skipping invalid negative values**

```
int [] data = {1, 2, 10, 5, -1, 5, 0};
int i = 0;
int sum = 0;
for (i = 0; i < data.length; i++)
{
    if (data[i] < 0)
        continue;
    sum += data[i];
}
System.out.println("Sum: " + sum);
```

```
% java SumNumsSkip
sum = 23
```

7

Conditional action from a set

- **Do something depending on a value value**
 - if-else if-else if... statements can get tedious

```
if (day == 1)
    monthStr = "Monday";
else if (day == 2)
    monthStr = "Tuesday";
else if (day == 3)
    monthStr = "Wednesday";
else if (day == 4)
    monthStr = "Thursday";
else if (day == 5)
    monthStr = "Friday";
else if (day == 6)
    monthStr = "Saturday";
else if (day == 7)
    monthStr = "Sunday";
else
    monthStr = "Invalid day!";
```

Set a String variable
monthStr to a string
according to the integer
value in the day
variable.

8

Conditional action from a set

- **switch statement**

- Works with: byte, short, char, int, enumerations
- Java 1.7: String

```
switch (day)
{
  case 1: monthStr = "Monday";   break;
  case 2: monthStr = "Tuesday";  break;
  case 3: monthStr = "Wednesday"; break;
  case 4: monthStr = "Thursday"; break;
  case 5: monthStr = "Friday";   break;
  case 6: monthStr = "Saturday";  break;
  case 7: monthStr = "Sunday";    break;
  default: monthStr = "Invalid day!"; break;
}
```

case block normally ends with a break

default block is optional, but if present executes if no other case matched. Like the else in an if-else if-else statement.

9

switch statement

```
final int NORTH = 0;
final int SOUTH = 1;
final int EAST = 2;
final int WEST = 3;

int direction = 0;

switch (direction)
{
  case NORTH:
    y--;
    System.out.println("Walking north");
    break;
  case SOUTH:
    y++;
    System.out.println("Walking south");
    break;
  case EAST:
    x++;
    System.out.println("Walking east");
    break;
  case WEST:
    x--;
    System.out.println("Walking west");
    break;
}
```

You can have as many statements as you want between case and break.

10


Buggy switch statement

```
final int NORTH = 0;
final int SOUTH = 1;
final int EAST = 2;
final int WEST = 3;
int direction = 0;

switch (direction)
{
  case NORTH:
    y--;
    System.out.println("Walking north");
  case SOUTH:
    y++;
    System.out.println("Walking south");
  case EAST:
    x++;
    System.out.println("Walking east");
  case WEST:
    x--;
    System.out.println("Walking west");
}
```

case block will fall through to next block if no break!

Output:
Walking north
Walking south
Walking east
Walking west



11

Falling through cases

```
int direction = 0;

switch (action)
{
  case NORTHWEST:
  case NORTHEAST:
  case NORTH:
    System.out.println("Heading northbound!");
    break;
  case SOUTHWEST:
  case SOUTHEAST:
  case SOUTH:
    System.out.println("Walking southbound!");
    break;
}
```

Sometimes falling through to next case block is what you want.
Easy way to do same thing for a set of discrete values.

Output:
Heading southbound

12

Summary

- **New loop constructs:**
 - Get out of loop early: **break**
 - Break is terminal for loop
 - Associated with closest enclosing loop
 - Skip rest of current loop: **continue**
 - Loop continues to execute
- **New conditional:**
 - When you have many choices: **switch**
 - Useful when you have many choices
 - Be careful to use break statement after each case