THE “WHILE” LOOP

CS302 – Introduction to Programming
University of Wisconsin – Madison
Lecture 7

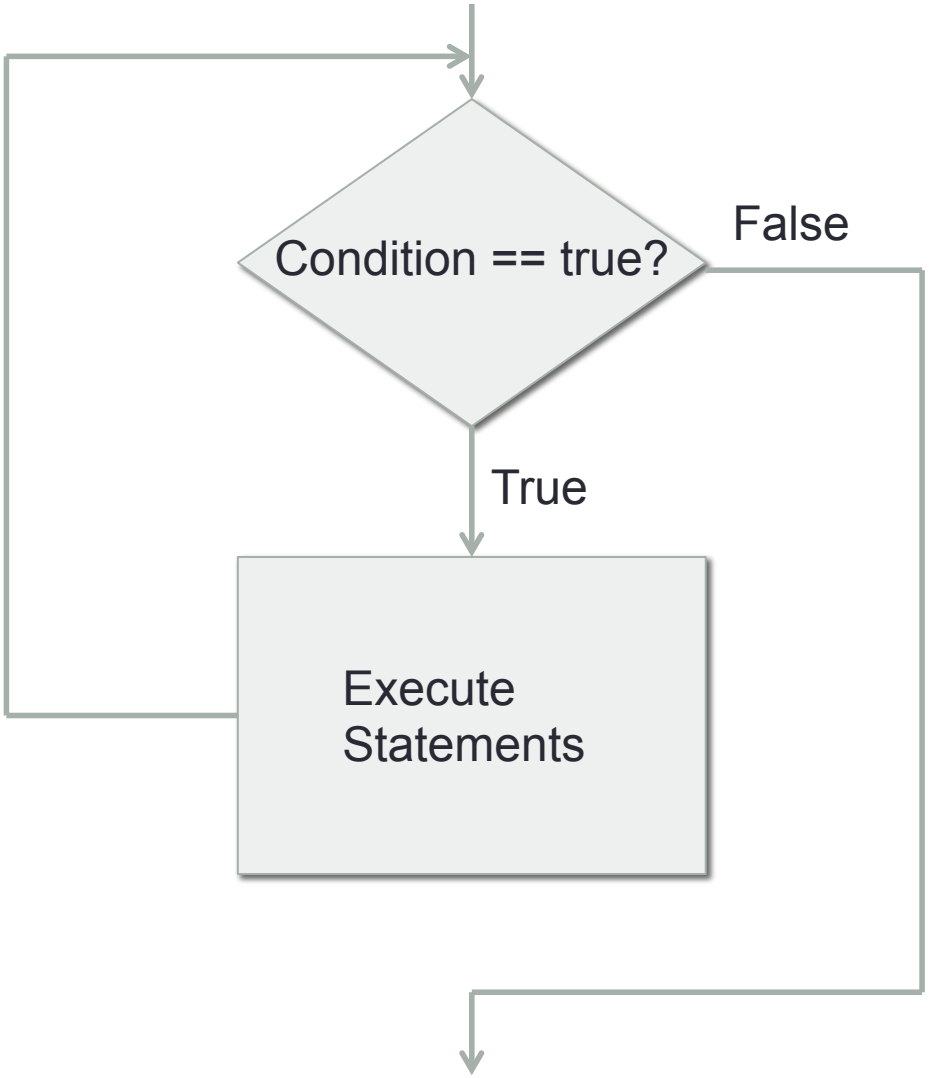
By Matthew Bernstein – matthewb@cs.wisc.edu

The “While” Loop

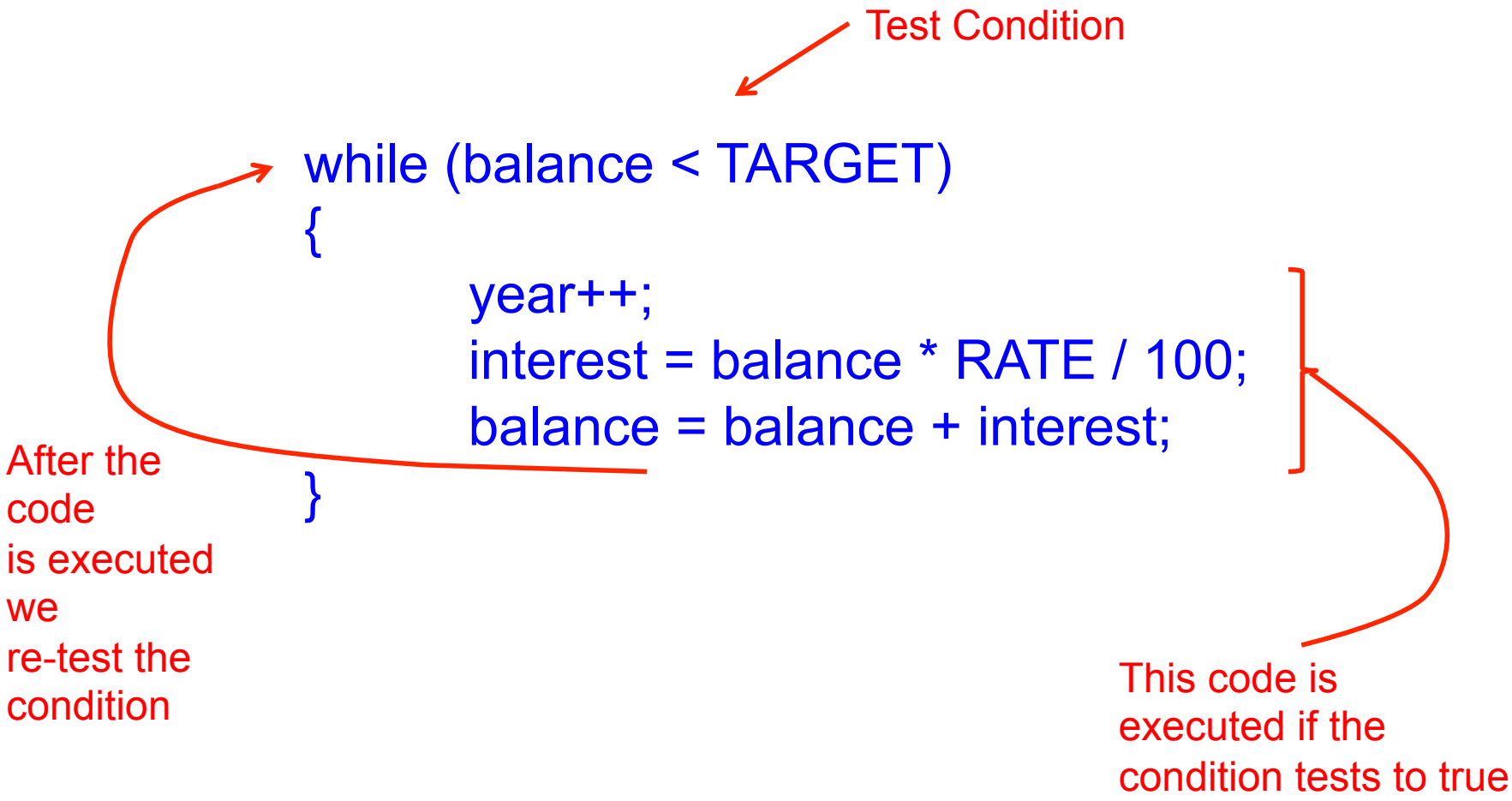
- A “while” loop is a block of code that will execute over and over again while some condition is met.
- The program first checks the condition. If the condition holds true, the program executes the statements within the while-loop. After executing the statements, the program goes back to the top and re-checks the condition.
- Form:

```
while (condition)
{
        statements
}
```

Flowchart of “While” Loop



Example


Test Condition
while (balance < TARGET)
{
year++;
interest = balance * RATE / 100;
balance = balance + interest;
}
After the code is executed we re-test the condition
This code is executed if the condition tests to true

What will happen here?

```
final double TARGET = 1000;
double balance = 2000;

while (balance < TARGET)
{
        year++;
        interest = balance * RATE / 100;
        balance = balance + interest;
}

System.out.println(year + " years to reach target");
```

What will happen here?

```
final double TARGET = 1000;
double balance = 50;

while (balance < TARGET)
{
        year++;
        interest = balance * RATE / 100;
}

System.out.println(year + " years to reach target");
```

The Break Statement

- The break statement breaks program execution out of the inner-most loop. The following program stops computing the balance if the years exceed 20.
- Example:

```
while (balance < TARGET)
{
        year++;
        if (year > 20)
        {
                System.out.println("Would take more than 20 years);
                break;
        }

        interest = balance * RATE / 100;
        balance = balance + interest;
}

System.println(years + " years");
```

The “return” statement

- The **return** statement ends execution within a method (it returns execution to whatever code called this method)
- So far, all of our programs have been written in the “**main**” method
- Thus, adding a return statement will essentially end the program (this is the case for the programs we have written so far)

Example

```
while (balance < TARGET)
{
        year++;
        if (year > 20)
        {
                System.out.println("Would take more than 20 years);
                return;
        }

        interest = balance * RATE / 100;
        balance = balance + interest;
}

System.println("Would take " + year + " years.");
```

Input Validation

- We loop until the user's input is valid (i.e. the conditional statement tests whether the user input was valid)
- Read about Scanner's hasNext(), hasNextInt(), hasNextDouble(), methods here:
  - http://docs.oracle.com/javase/1.5.0/docs/api/java/util/Scanner.html

Programming Exercise

- Change your temperature converter program such that when the user enters invalid input, your program prompts the user to enter new input.
- Example Program Execution:

```
Input Units (Enter “F” or “C”): V
Invalid input, please try again: 3.0
Invalid input, please try again: F
Output Units: C
Input Temperature: slkdjlfksdlf
Invalid input, please try again: 0
0.0 F =  -17.7778 C
```

Back to "While" Loops Problem-Solving: Hand-Tracing

Consider the example:

```
int year = 1;
while (balance < TARGET)
{
        year++;

        balance = balance * ( 1 + RATE / 100);
}

System.println("Would take " + year + " years.");
```

- Should year start at 0 or 1?
- Should the condition use "balance < TARGET" or "balance <= TARGET"?

Hand Tracing

- Create a table where each column corresponds to a variable and each row corresponds to an iteration of the loop.
- Let's say, our balance starts at $100, our interest rate is %50, and our target is $200
- Resulting Table:

year	balance
1	100
2	150
3	225

- This would output 3 years, even though it only took us 2 years to reach $200. We should start "year" at 0, not 1.

Hand Tracing

- Now let us assume our target is $225 (our initial balance is still $100 and our interest rate is still %50)

year	balance
0	100
1	150
2	225

- At the 3rd evaluation of our conditional statement, (when year = 2 and balance = 225), we see that we have reached our target and therefore we should not execute the code inside the loop again. Thus, we should use "balance < TARGET" instead of "balance <= TARGET"

Hand-Tracing

What will be the final sum after the following code executes:

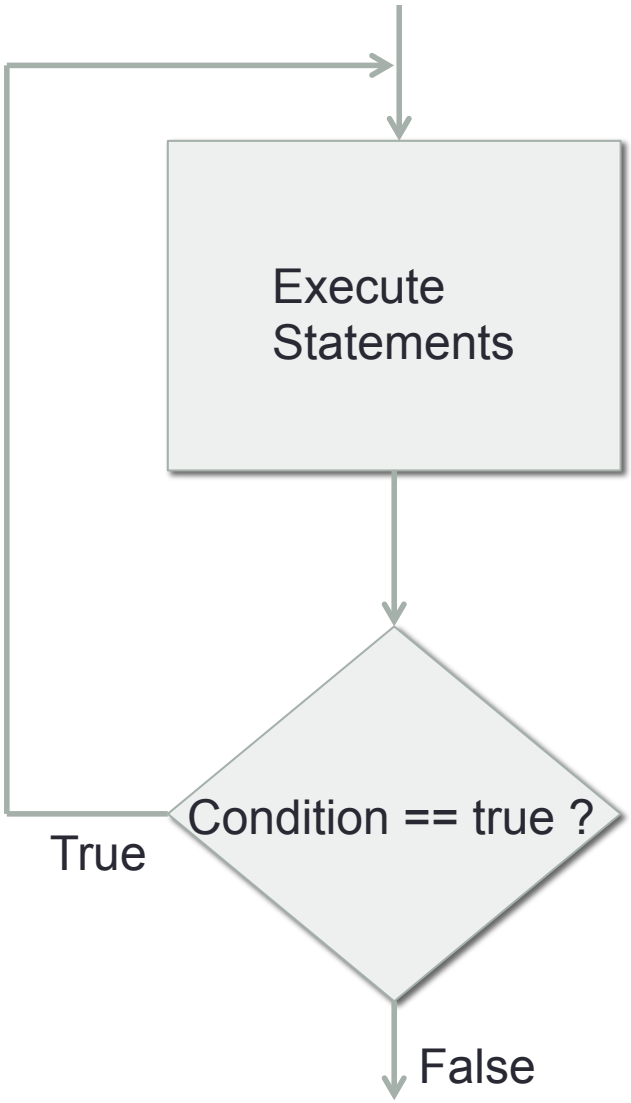
```
int n = 1729;
int sum = 0;
int digit;

while (n > 0)
{
        digit = n %10;
        sum = sum + digit;
        n = n / 10;
}
```

The “Do” Loop

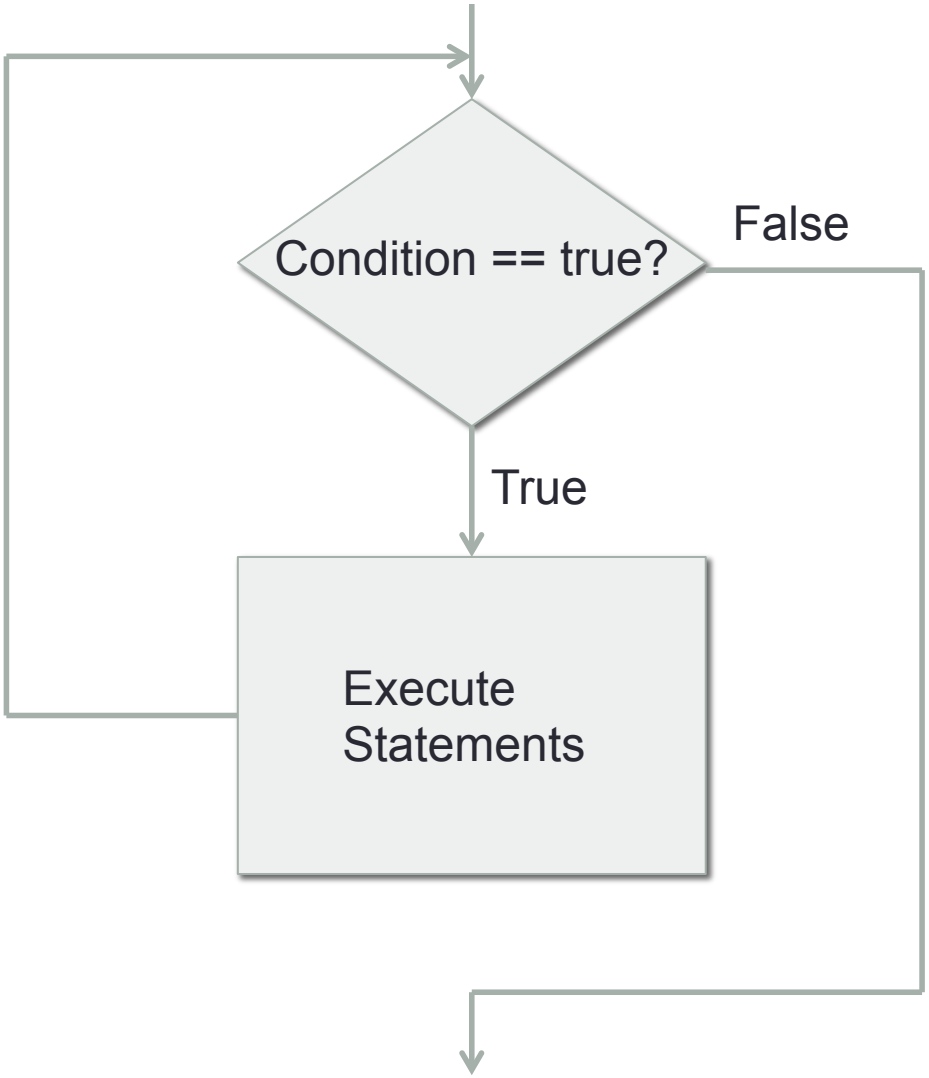
- Sometimes you want to execute the body of a loop at least once and perform the loop test after the body is executed. To do this, we use the “do” loop:

```
do
{
        statements
}
while (condition)
```

Flowchart of the “Do” Loop



Compared to flowchart of “While” Loop



Example

```
int value;

do
{
        System.out.print("Enter an integer < 100: ");
        value = in.nextInt();  // "in" is a Scanner object
}
while (value >= 100);
```

Programming Exercise

- Adjust your Rock Paper Scissors program to allow the user to play multiple rounds against the computer. The program should keep a score for the number of wins by the player and computer. After each round, the program should output the scores.
- When the user enters “q” the program should quit. When the user quits, the program should output the final score.
- The program should now validate that the user entered either “rock”, “paper”, “scissors” or “q”.

Example Execution

What hand will you throw? (Input either "rock", "paper", "scissors", or "q" to quit):
rock
Computer threw scissors, you threw rock
You won! (Score: Player 1, Computer 0)

What hand will you throw? (Input either "rock", "paper", "scissors", or "q" to quit):
paper
Computer threw scissors, you threw paper
You lost! (Score: Player 1, Computer 1)

What hand will you throw? (Input either "rock", "paper", "scissors", or "q" to quit):
rock
Computer threw rock, you threw rock
Tie! (Score: Player 1, Computer 1)

What hand will you throw? (Input either "rock", "paper", "scissors", or "q" to quit):
q
Thanks for playing! (Final Score: Player 1, Computer 1)

Let’s Code…

- Let’s start building our Rock, Paper, Scissors game

Cool CS Link of the Day

- Google Maps 3D Buildings:
- http://www.youtube.com/watch?v=N6Douyfa7I8