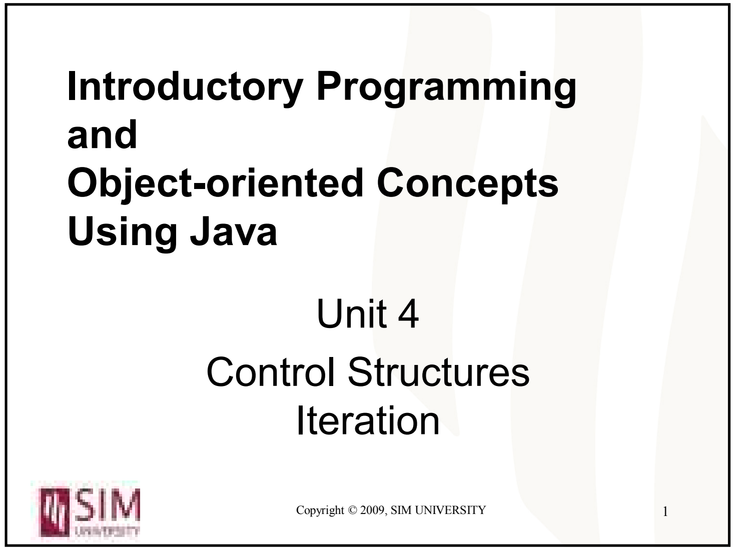
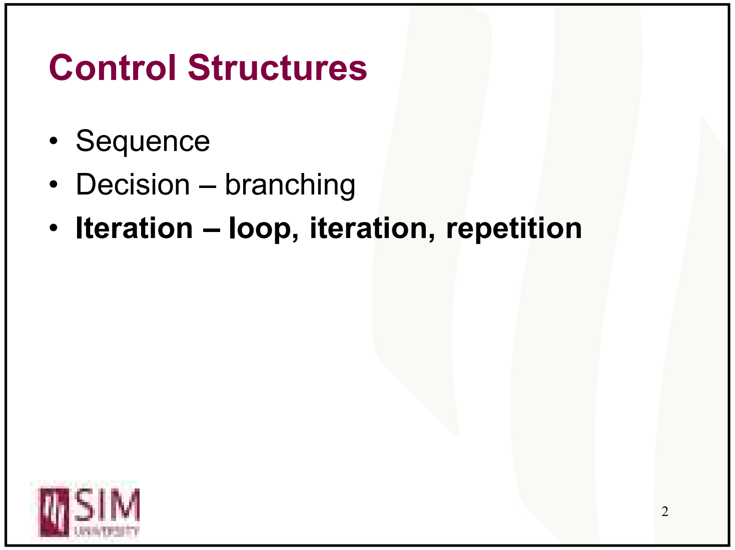


Introductory Programming and Object-oriented Concepts Using Java

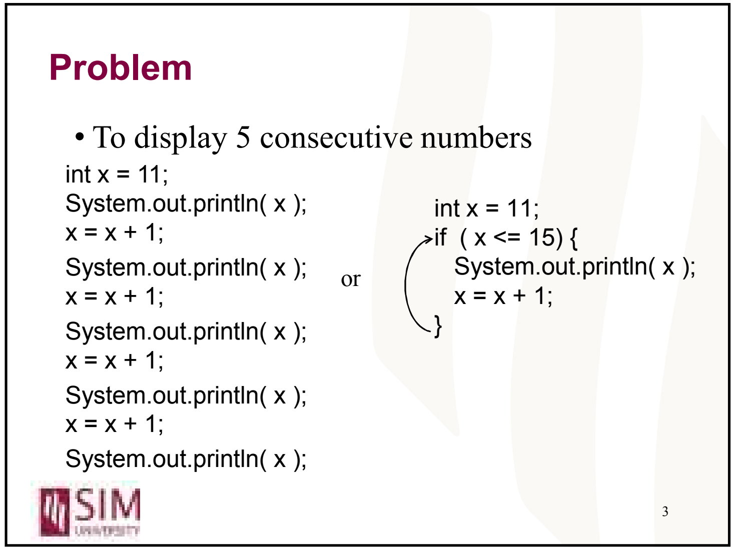
Unit 4

Control Structures
Iteration



Control Structures

- Sequence
- Decision – branching
- **Iteration – loop, iteration, repetition**



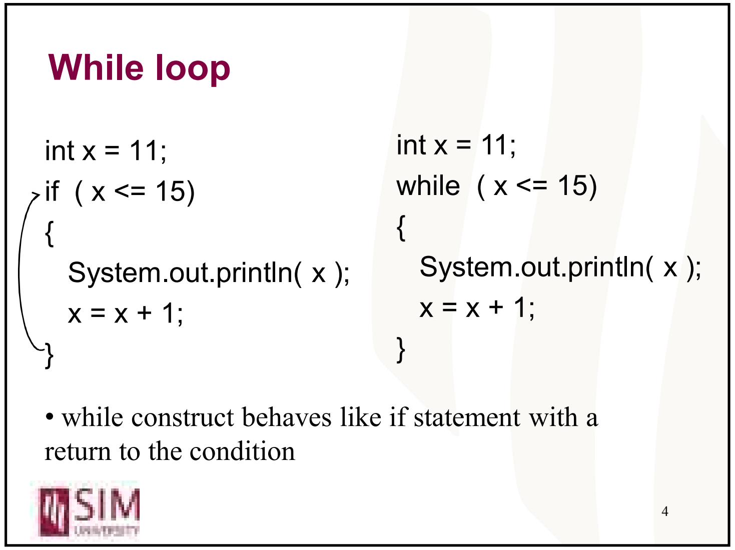
Problem

- To display 5 consecutive numbers

```
int x = 11;
System.out.println( x );
x = x + 1;
System.out.println( x );
x = x + 1;
System.out.println( x );
x = x + 1;
System.out.println( x );
x = x + 1;
System.out.println( x );
```

or

```
int x = 11;
if  ( x <= 15) {
   System.out.println( x );
   x = x + 1;
}
```

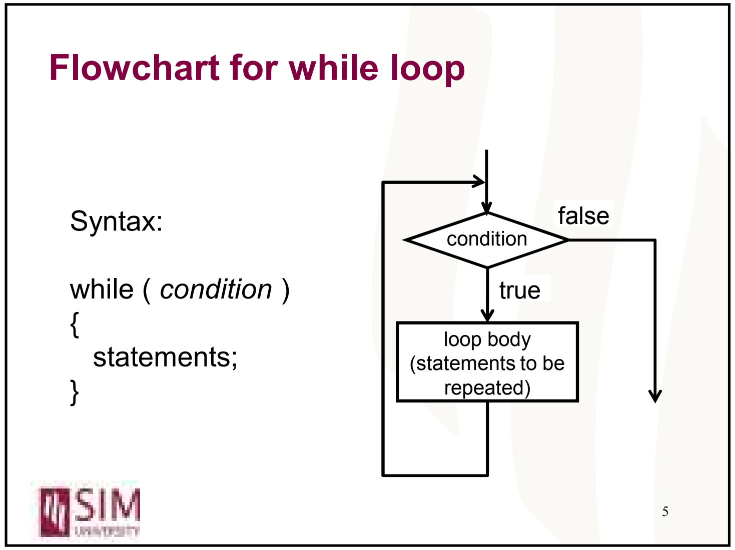


While loop

```
int x = 11;
if  ( x <= 15)
{
   System.out.println( x );
   x = x + 1;
}
```

```
int x = 11;
while  ( x <= 15)
{
   System.out.println( x );
   x = x + 1;
}
```

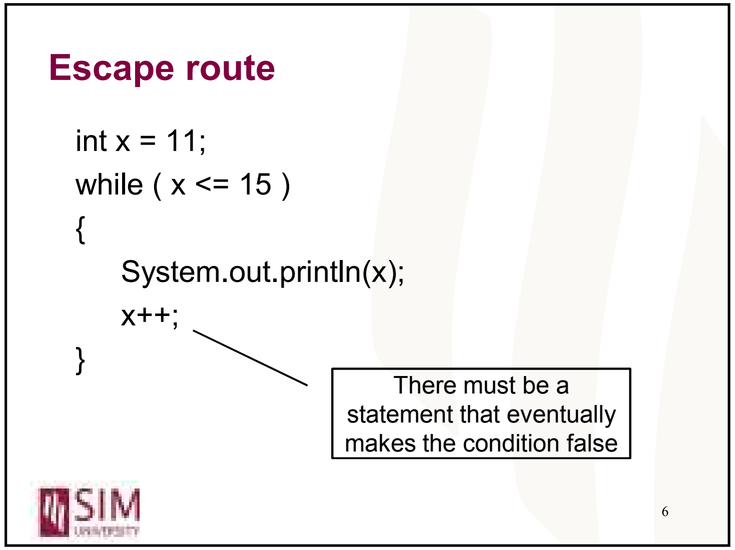
- while construct behaves like if statement with a return to the condition



Flowchart for while loop

Syntax:

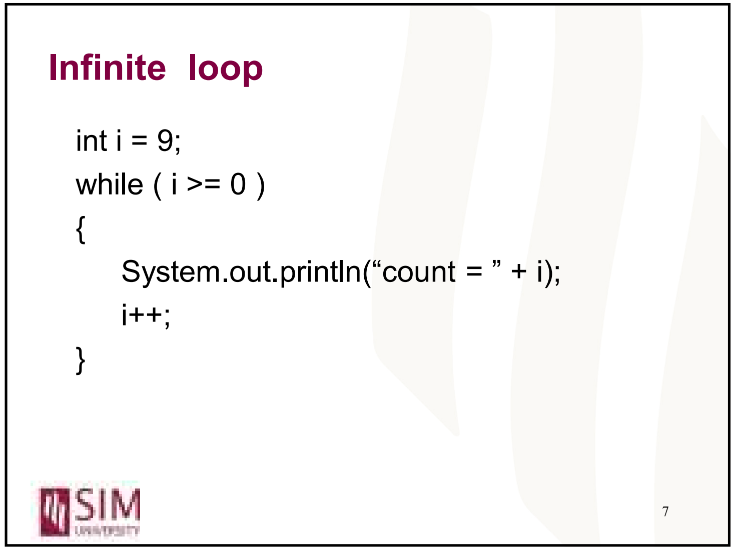
```
while ( condition )
{
  statements;
}
```



Escape route

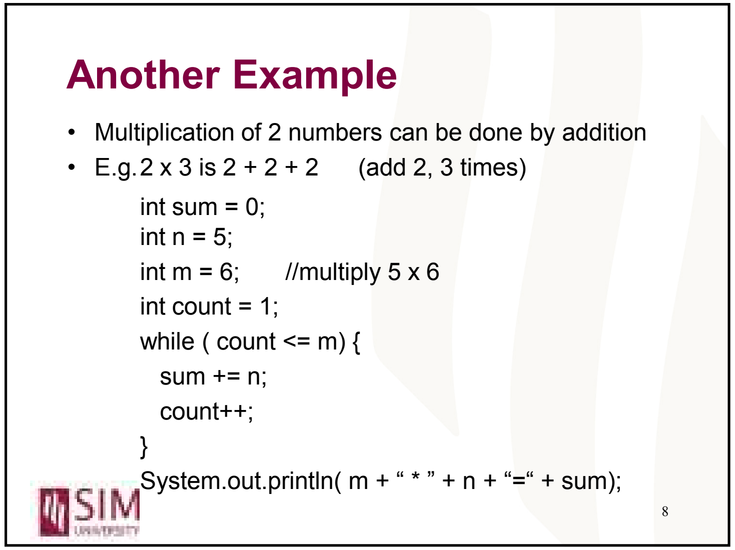
```
int x = 11;
while ( x <= 15 )
{
    System.out.println(x);
    x++;
}
```

There must be a statement that eventually makes the condition false



Infinite loop

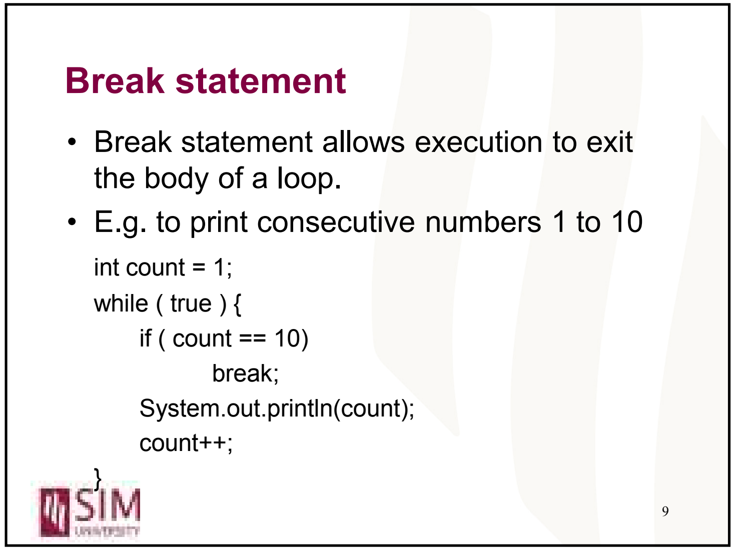
```
int i = 9;
while ( i >= 0 )
{
    System.out.println(“count = ” + i);
    i++;
}
```



Another Example

- Multiplication of 2 numbers can be done by addition
- E.g. 2 x 3 is 2 + 2 + 2   (add 2, 3 times)

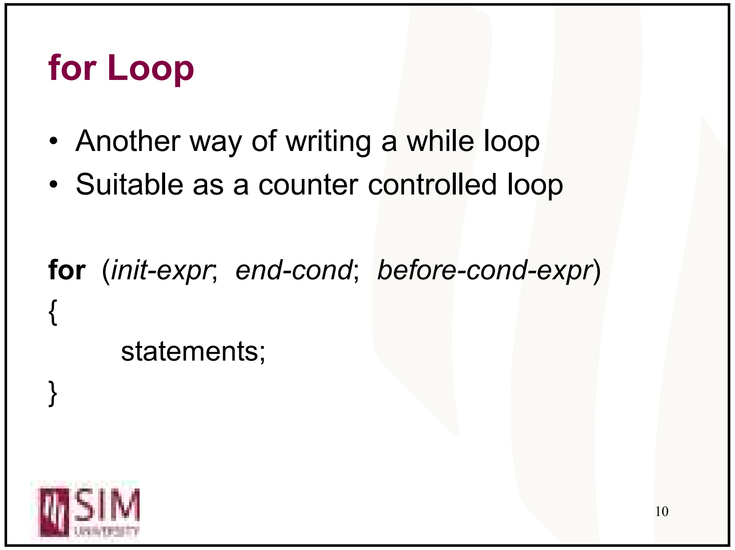
```
int sum = 0;
int n = 5;
int m = 6;      //multiply 5 x 6
int count = 1;
while ( count <= m) {
  sum += n;
  count++;
}
System.out.println( m + “ * ” + n + “=“ + sum);
```



Break statement

- Break statement allows execution to exit the body of a loop.
- E.g. to print consecutive numbers 1 to 10

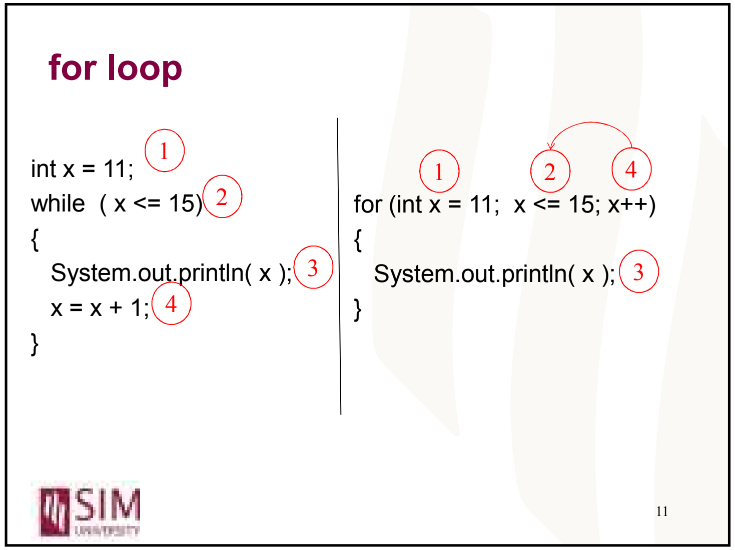
```
int count = 1;
while ( true ) {
    if ( count == 10)
          break;
    System.out.println(count);
    count++;
}
```



for Loop

- Another way of writing a while loop
- Suitable as a counter controlled loop

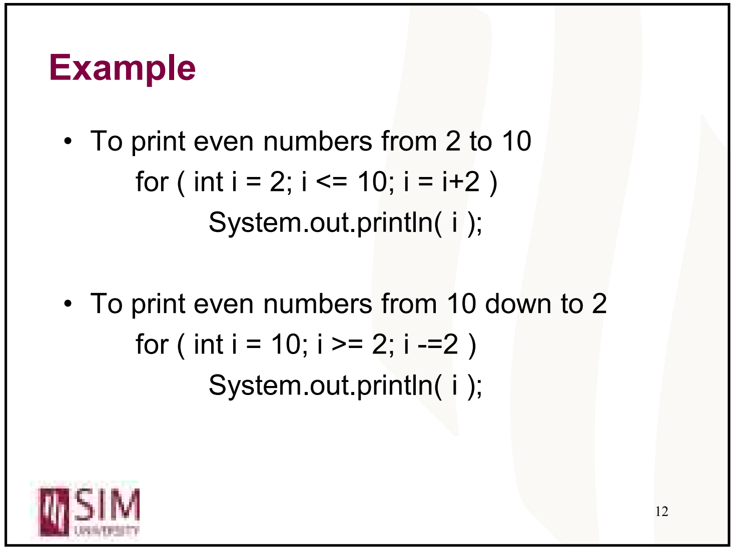
for (*init-expr*; *end-cond*; *before-cond-expr*)
{
    statements;
}



for loop

```
int x = 11; ①
while  ( x <= 15) ②
{
  System.out.println( x ); ③
  x = x + 1; ④
}
```

```
for (int x = 11; ① x <= 15; ② x++) ④
{
  System.out.println( x ); ③
}
```



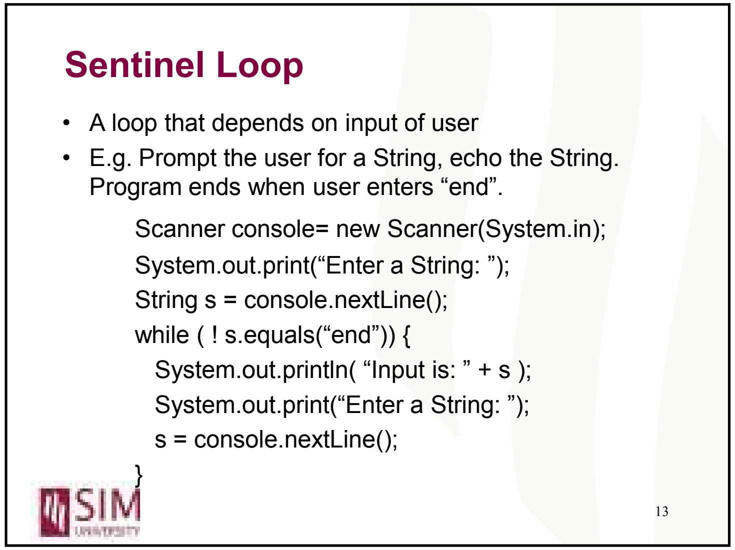
Example

- To print even numbers from 2 to 10

```
    for ( int i = 2; i <= 10; i = i+2 )
          System.out.println( i );
```

- To print even numbers from 10 down to 2

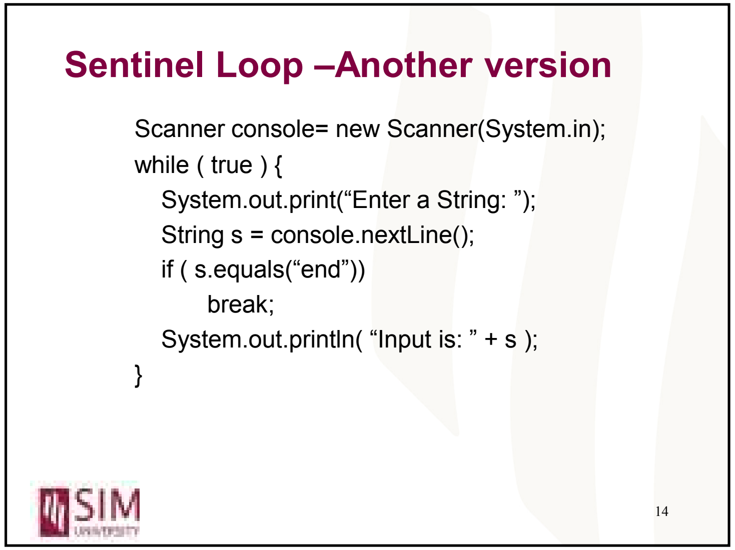
```
    for ( int i = 10; i >= 2; i -=2 )
          System.out.println( i );
```



Sentinel Loop

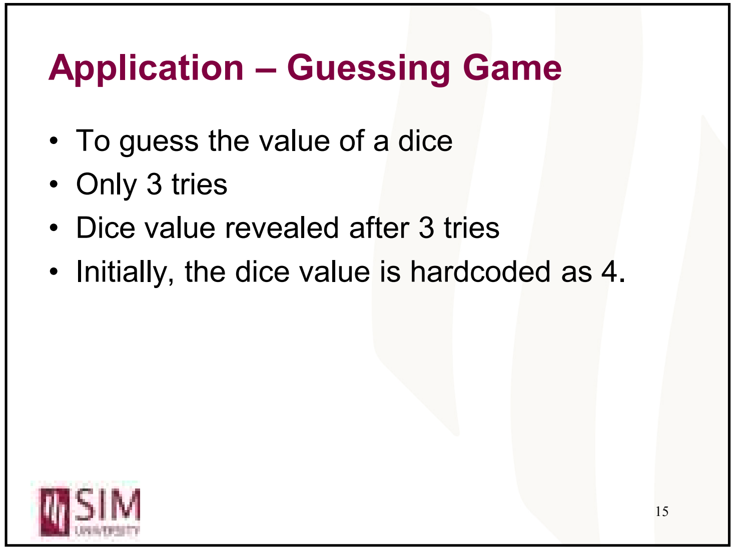
- A loop that depends on input of user
- E.g. Prompt the user for a String, echo the String. Program ends when user enters “end”.

```
Scanner console= new Scanner(System.in);
System.out.print(“Enter a String: ”);
String s = console.nextLine();
while ( ! s.equals(“end”)) {
   System.out.println( “Input is: ” + s );
   System.out.print(“Enter a String: ”);
   s = console.nextLine();
}
```



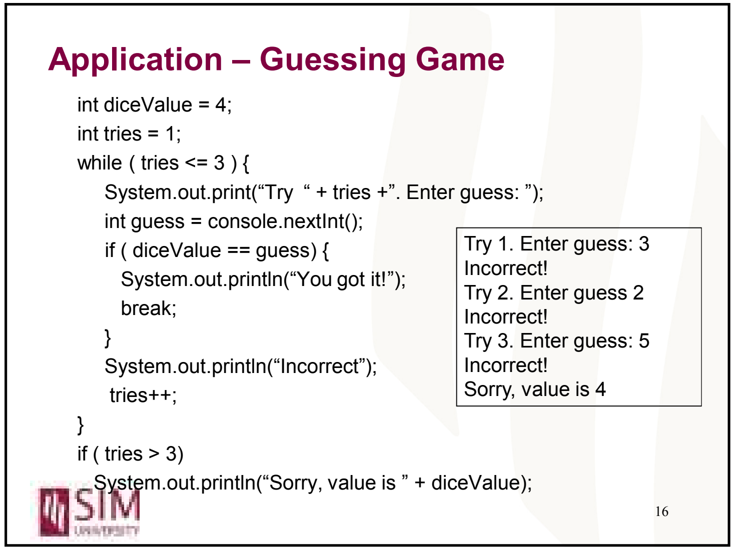
Sentinel Loop –Another version

```
Scanner console= new Scanner(System.in);
while ( true ) {
   System.out.print(“Enter a String: ”);
   String s = console.nextLine();
   if ( s.equals(“end”))
        break;
   System.out.println( “Input is: ” + s );
}
```



Application – Guessing Game

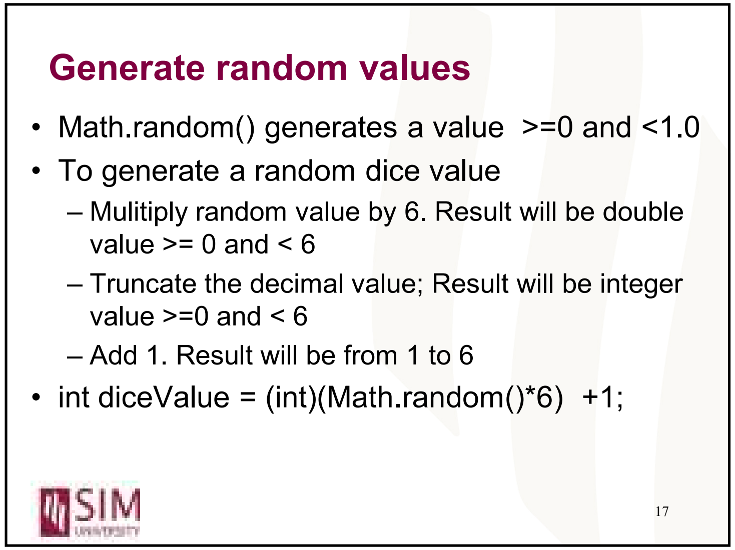
- To guess the value of a dice
- Only 3 tries
- Dice value revealed after 3 tries
- Initially, the dice value is hardcoded as 4.



Application – Guessing Game

```
int diceValue = 4;
int tries = 1;
while ( tries <= 3 ) {
    System.out.print(“Try  “ + tries +”. Enter guess: ”);
    int guess = console.nextInt();
    if ( diceValue == guess) {
      System.out.println(“You got it!”);
      break;
    }
    System.out.println(“Incorrect”);
     tries++;
}
if ( tries > 3)
  System.out.println(“Sorry, value is ” + diceValue);
```

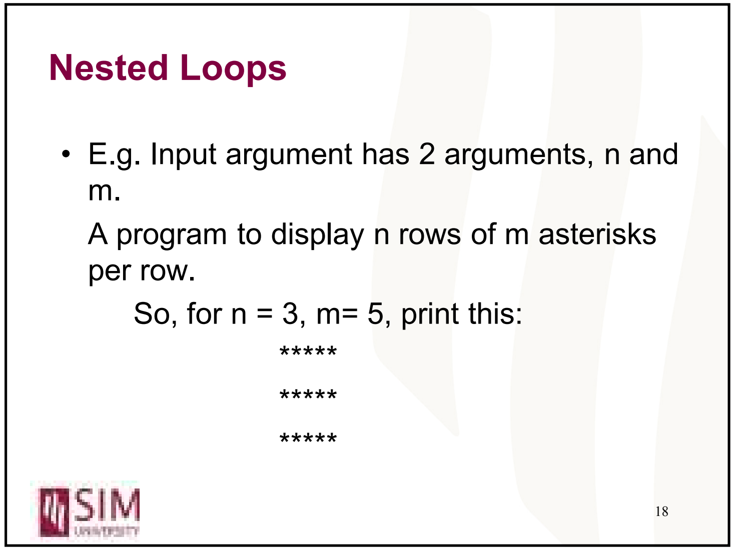
```
Try 1. Enter guess: 3
Incorrect!
Try 2. Enter guess 2
Incorrect!
Try 3. Enter guess: 5
Incorrect!
Sorry, value is 4
```



Generate random values

- Math.random() generates a value >=0 and <1.0
- To generate a random dice value
  - Mulitiply random value by 6. Result will be double value >= 0 and < 6
  - Truncate the decimal value; Result will be integer value >=0 and < 6
  - Add 1. Result will be from 1 to 6
- int diceValue = (int)(Math.random()*6) +1;

17



Nested Loops

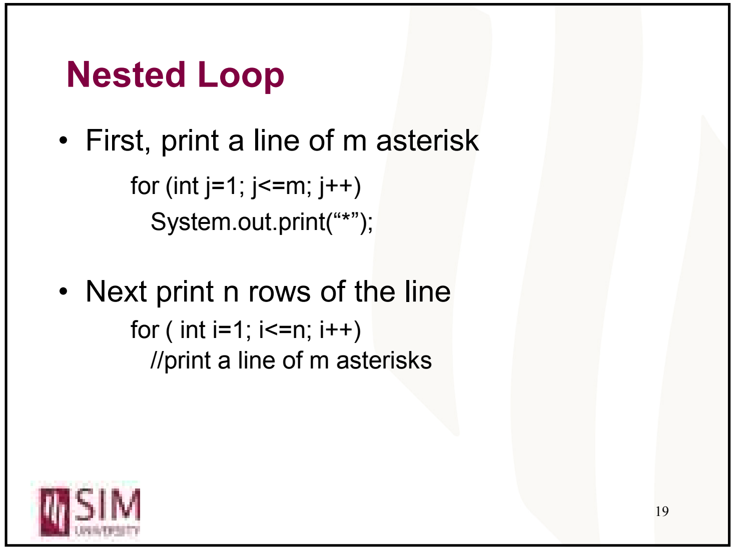
- E.g. Input argument has 2 arguments, n and m.

  A program to display n rows of m asterisks per row.

  So, for n = 3, m= 5, print this:

```
*****
*****
*****
```

18



Nested Loop

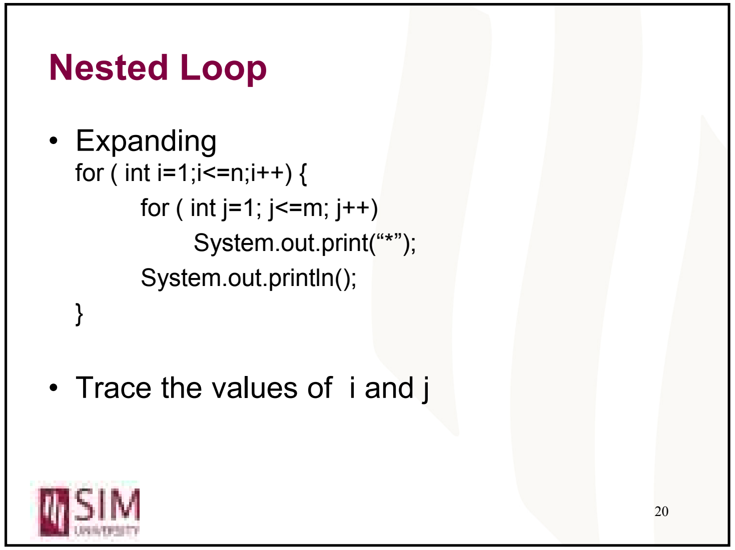
- First, print a line of m asterisk

```
for (int j=1; j<=m; j++)
  System.out.print("*");
```

- Next print n rows of the line

```
for ( int i=1; i<=n; i++)
  //print a line of m asterisks
```

19



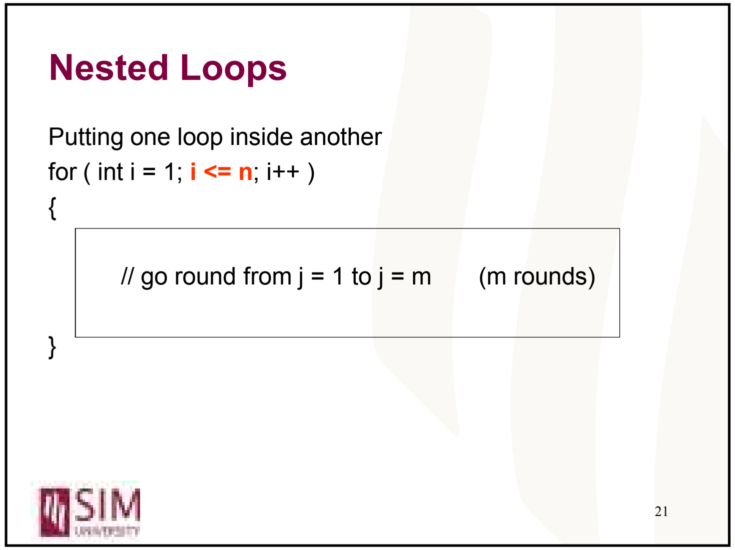
Nested Loop

- Expanding

```
for ( int i=1;i<=n;i++) {
      for ( int j=1; j<=m; j++)
            System.out.print("*");
      System.out.println();
}
```

- Trace the values of i and j

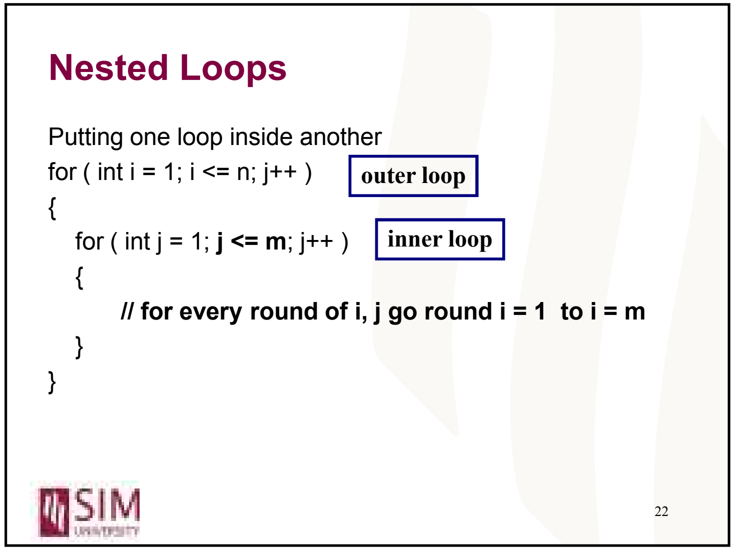
20



Nested Loops

Putting one loop inside another

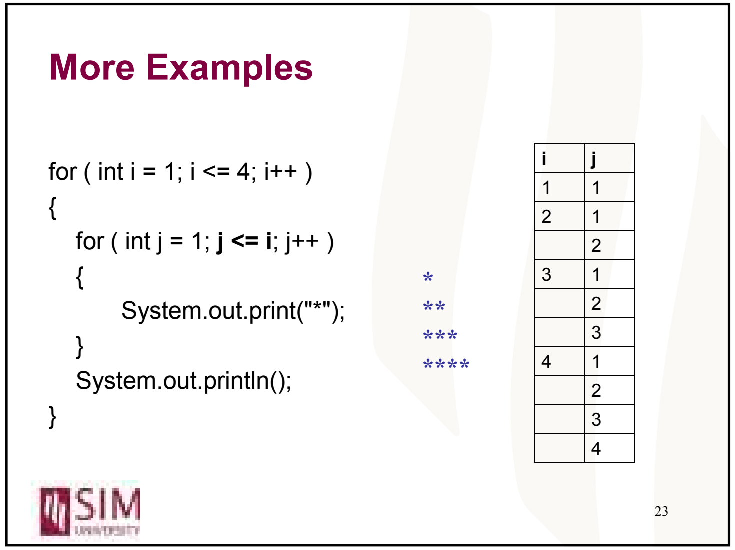
```
for ( int i = 1; i <= n; i++ )
{
        // go round from j = 1 to j = m        (m rounds)
}
```



Nested Loops

Putting one loop inside another

```
for ( int i = 1; i <= n; j++ )      outer loop
{
   for ( int j = 1; j <= m; j++ )    inner loop
   {
        // for every round of i, j go round i = 1  to i = m
   }
}
```

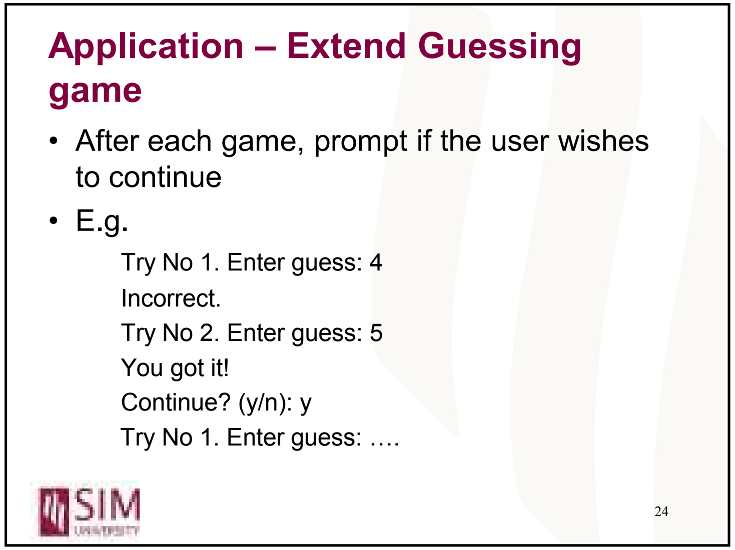


More Examples

```
for ( int i = 1; i <= 4; i++ )
{
   for ( int j = 1; j <= i; j++ )
   {
        System.out.print("*");
   }
   System.out.println();
}
```

```
*
**
***
****
```

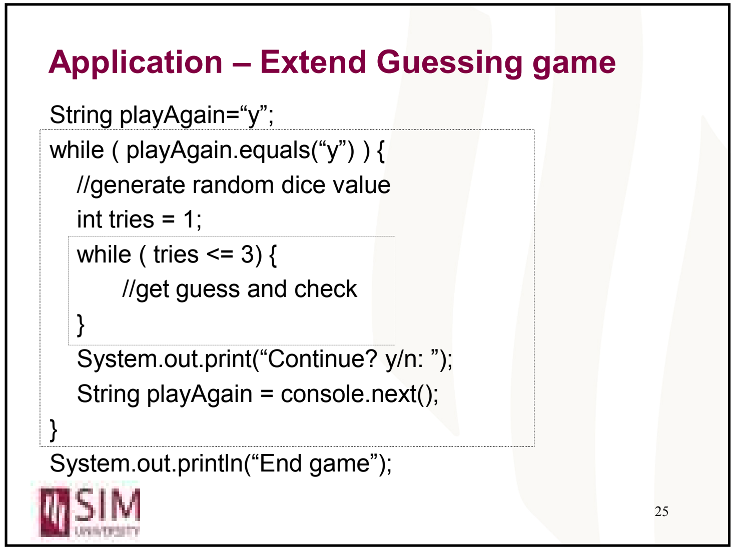
i	j
1	1
2	1
	2
3	1
	2
	3
4	1
	2
	3
	4



Application – Extend Guessing game

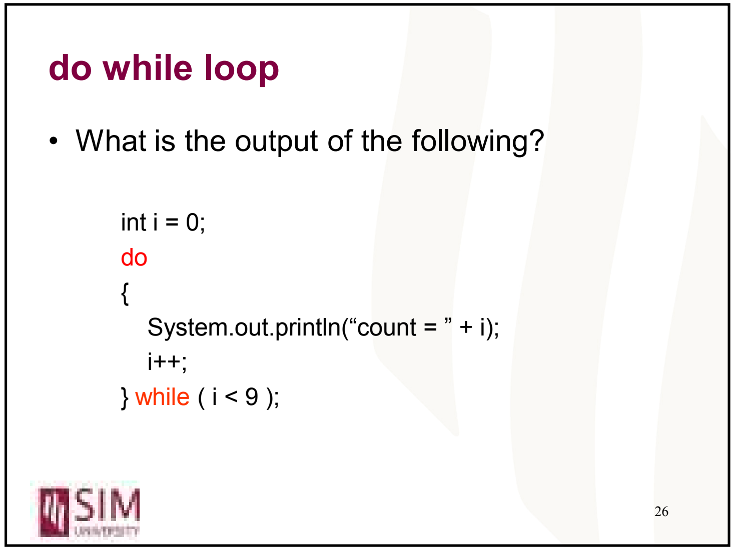
- After each game, prompt if the user wishes to continue
- E.g.

```
Try No 1. Enter guess: 4
Incorrect.
Try No 2. Enter guess: 5
You got it!
Continue? (y/n): y
Try No 1. Enter guess: ….
```



Application – Extend Guessing game

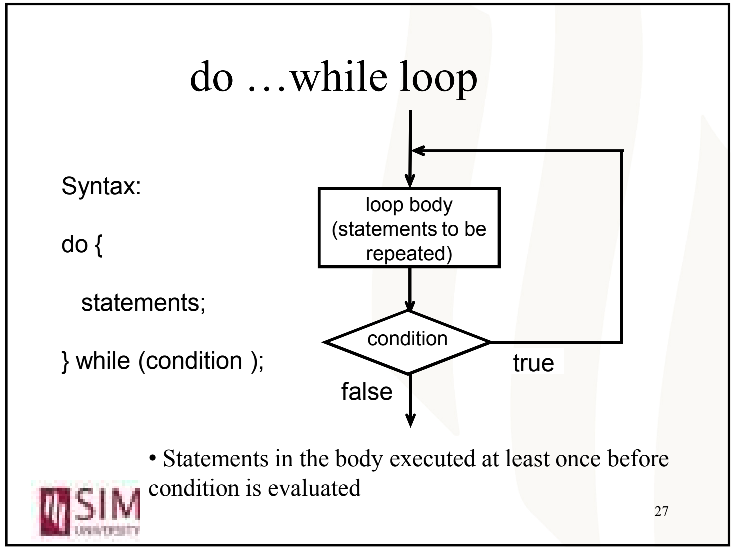
```
String playAgain=“y”;
while ( playAgain.equals(“y”) ) {
   //generate random dice value
   int tries = 1;
   while ( tries <= 3) {
        //get guess and check
   }
   System.out.print(“Continue? y/n: ”);
   String playAgain = console.next();
}
System.out.println(“End game”);
```



do while loop

- What is the output of the following?

```
int i = 0;
do
{
   System.out.println(“count = ” + i);
   i++;
} while ( i < 9 );
```

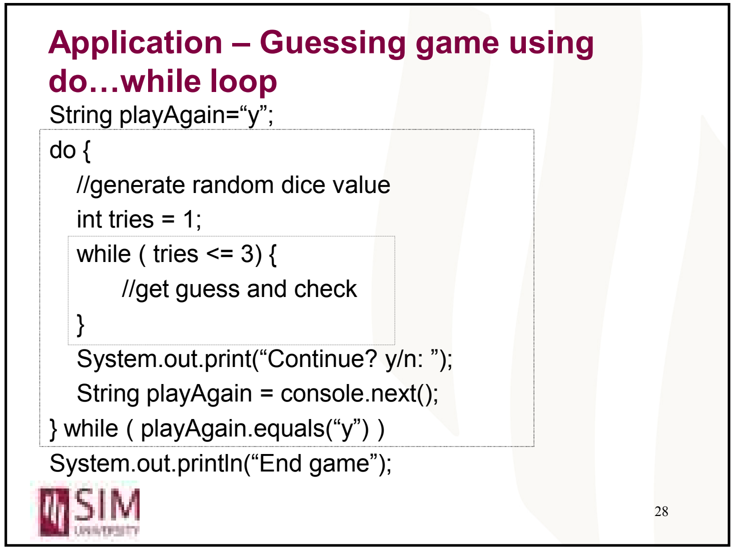


do …while loop

Syntax:

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

- Statements in the body executed at least once before condition is evaluated



Application – Guessing game using do…while loop

```
String playAgain=“y”;
do {
   //generate random dice value
   int tries = 1;
   while ( tries <= 3) {
        //get guess and check
   }
   System.out.print(“Continue? y/n: ”);
   String playAgain = console.next();
} while ( playAgain.equals(“y”) )
System.out.println(“End game”);
```