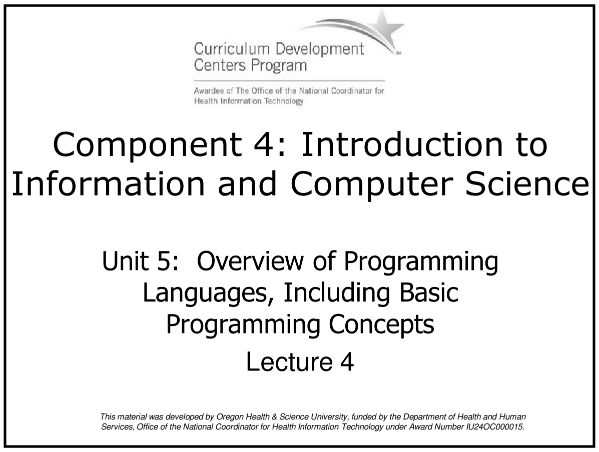


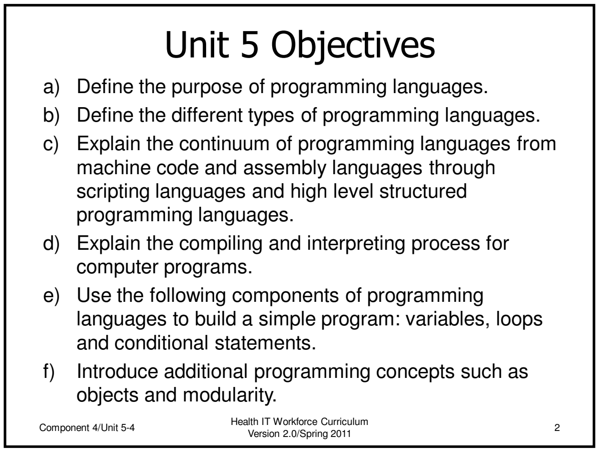
Curriculum Development Centers Program
Awardee of The Office of the National Coordinator for Health Information Technology

Component 4: Introduction to Information and Computer Science

Unit 5: Overview of Programming Languages, Including Basic Programming Concepts

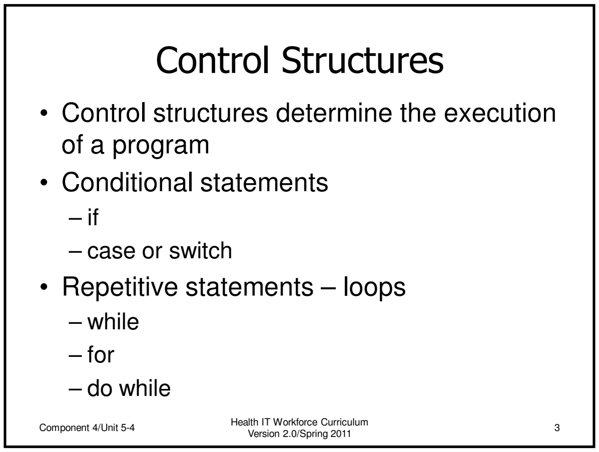
Lecture 4

This material was developed by Oregon Health & Science University, funded by the Department of Health and Human Services, Office of the National Coordinator for Health Information Technology under Award Number IU24OC000015.



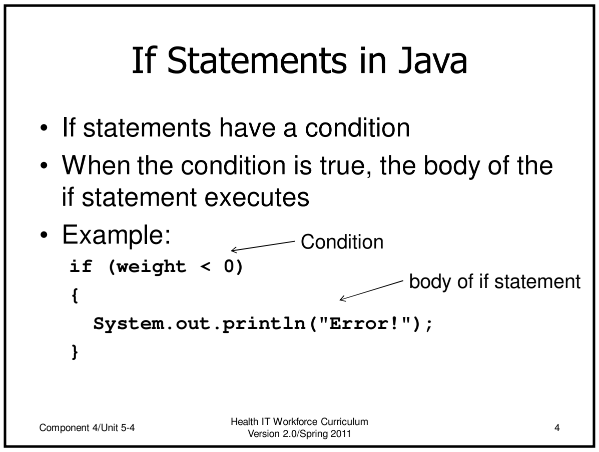
Unit 5 Objectives

a) Define the purpose of programming languages.
b) Define the different types of programming languages.
c) Explain the continuum of programming languages from machine code and assembly languages through scripting languages and high level structured programming languages.
d) Explain the compiling and interpreting process for computer programs.
e) Use the following components of programming languages to build a simple program: variables, loops and conditional statements.
f) Introduce additional programming concepts such as objects and modularity.



Control Structures

- Control structures determine the execution of a program
- Conditional statements
  - if
  - case or switch
- Repetitive statements – loops
  - while
  - for
  - do while



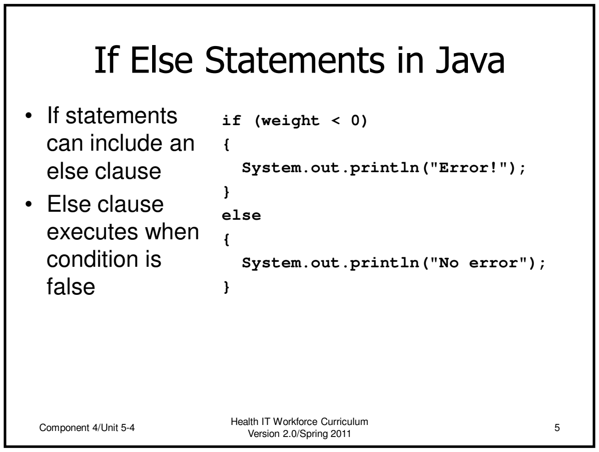
If Statements in Java

- If statements have a condition
- When the condition is true, the body of the if statement executes
- Example:

Condition

body of if statement

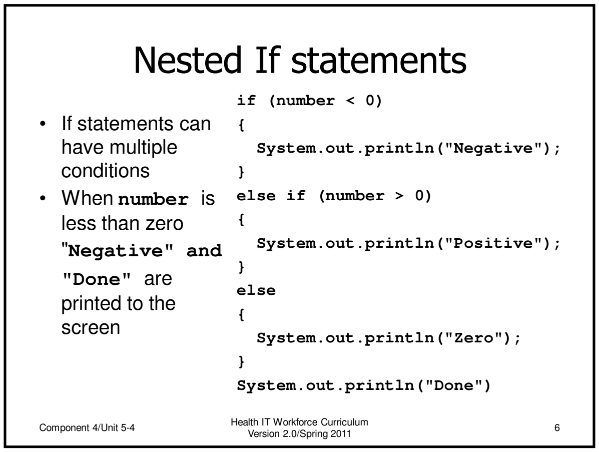
```
if (weight < 0)
{
  System.out.println("Error!");
}
```



If Else Statements in Java

- If statements can include an else clause
- Else clause executes when condition is false

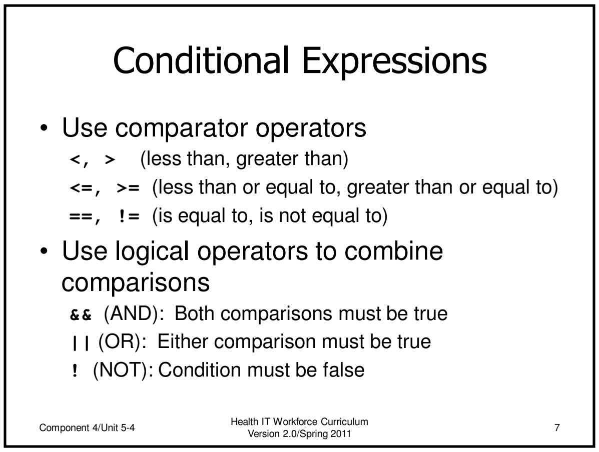
```
if (weight < 0)
{
  System.out.println("Error!");
}
else
{
  System.out.println("No error");
}
```



Nested If statements

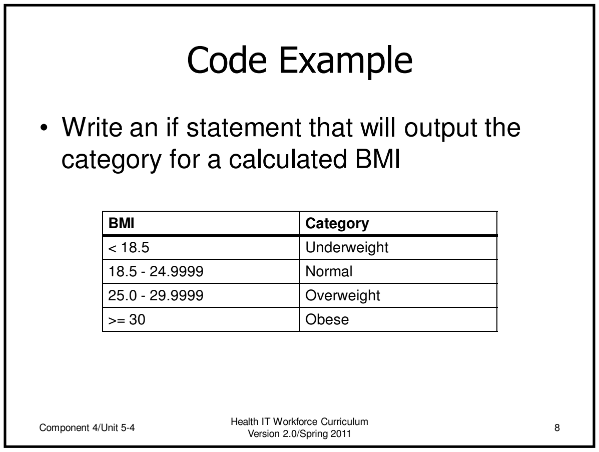
- If statements can have multiple conditions
- When `number` is less than zero `"Negative" and "Done"` are printed to the screen

```
if (number < 0)
{
  System.out.println("Negative");
}
else if (number > 0)
{
  System.out.println("Positive");
}
else
{
  System.out.println("Zero");
}
System.out.println("Done")
```



Conditional Expressions

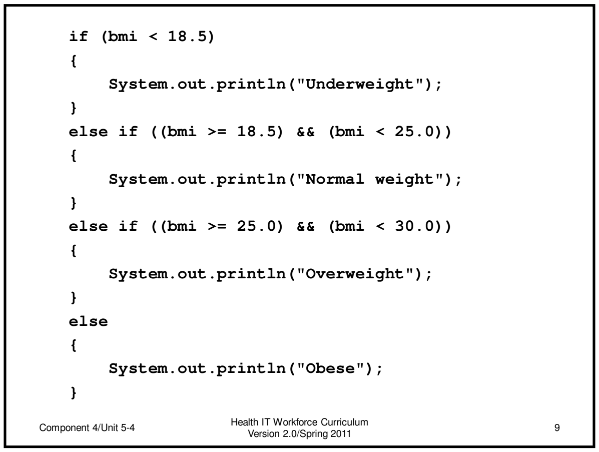
- Use comparator operators
  - `<, >` (less than, greater than)
  - `<=, >=` (less than or equal to, greater than or equal to)
  - `==, !=` (is equal to, is not equal to)
- Use logical operators to combine comparisons
  - `&&` (AND): Both comparisons must be true
  - `||` (OR): Either comparison must be true
  - `!` (NOT): Condition must be false



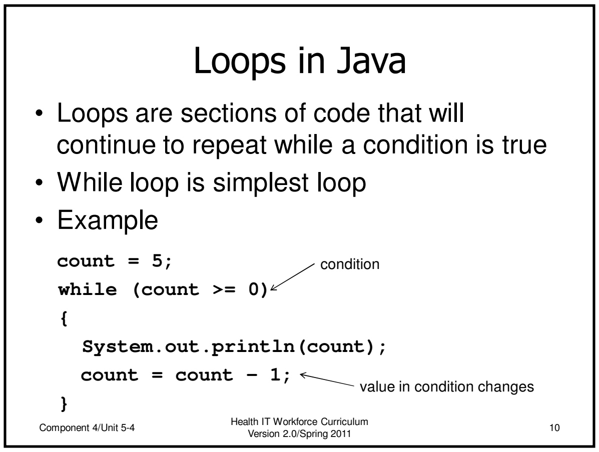
Code Example

- Write an if statement that will output the category for a calculated BMI

BMI	Category
< 18.5	Underweight
18.5 - 24.9999	Normal
25.0 - 29.9999	Overweight
>= 30	Obese



```
if (bmi < 18.5)
{
    System.out.println("Underweight");
}
else if ((bmi >= 18.5) && (bmi < 25.0))
{
    System.out.println("Normal weight");
}
else if ((bmi >= 25.0) && (bmi < 30.0))
{
    System.out.println("Overweight");
}
else
{
    System.out.println("Obese");
}
```



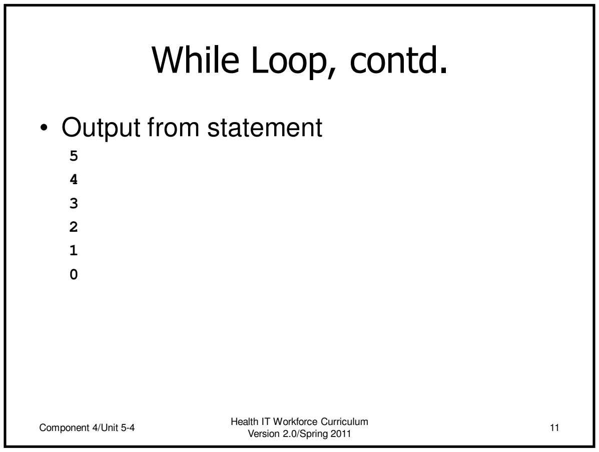
Loops in Java

- Loops are sections of code that will continue to repeat while a condition is true
- While loop is simplest loop
- Example

```
count = 5;
while (count >= 0)
{
  System.out.println(count);
  count = count – 1;
}
```

condition

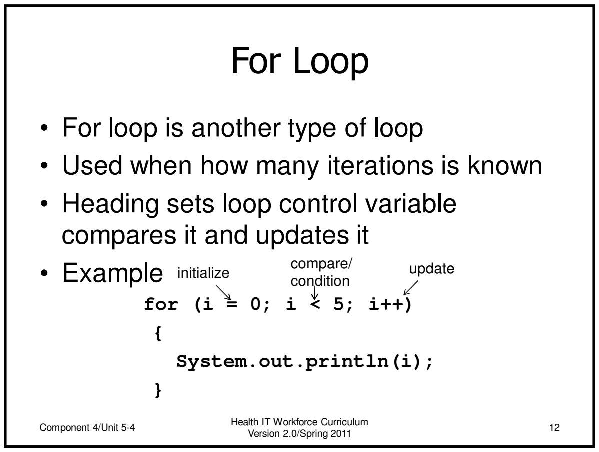
value in condition changes



While Loop, contd.

- Output from statement

```
5
4
3
2
1
0
```

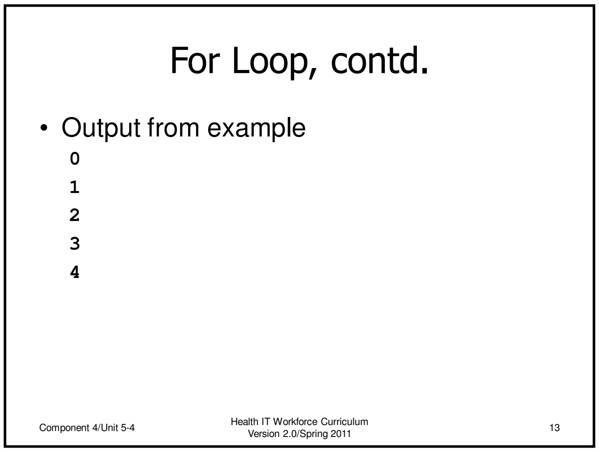


For Loop

- For loop is another type of loop
- Used when how many iterations is known
- Heading sets loop control variable compares it and updates it
- Example

initialize compare/ condition update

```
for (i = 0; i < 5; i++)
{
  System.out.println(i);
}
```

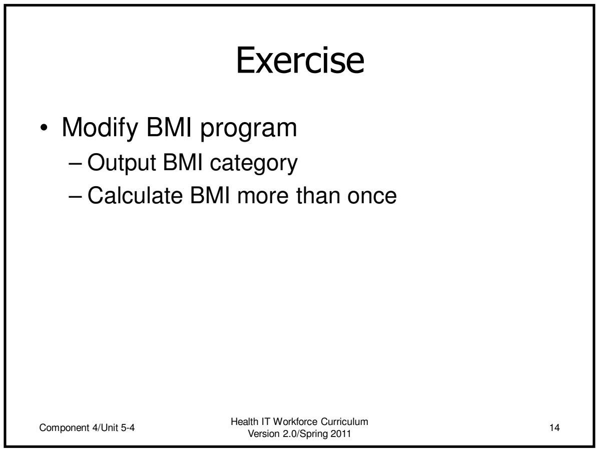


For Loop, contd.

- Output from example

```
0
1
2
3
4
```

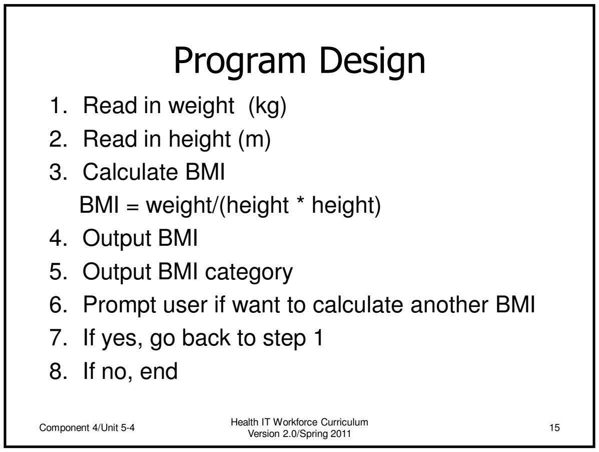
Component 4/Unit 5-4 Health IT Workforce Curriculum Version 2.0/Spring 2011 13



Exercise

- Modify BMI program
  - Output BMI category
  - Calculate BMI more than once

Component 4/Unit 5-4 Health IT Workforce Curriculum Version 2.0/Spring 2011 14

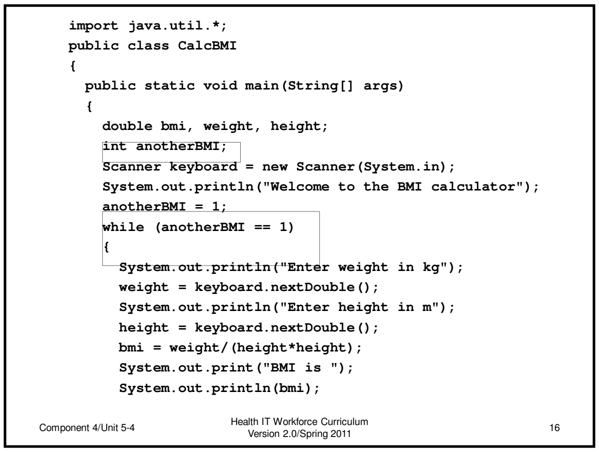


Program Design

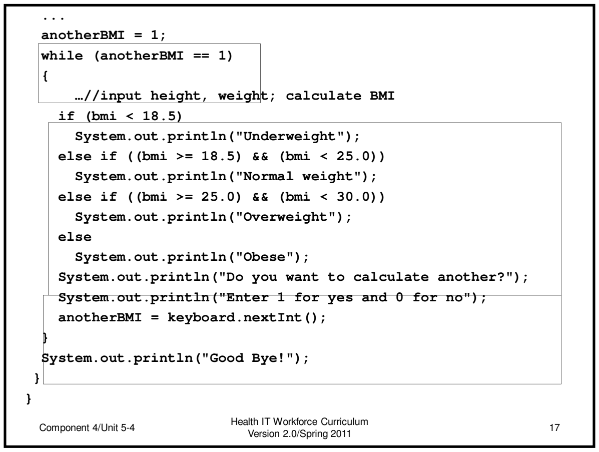
1. Read in weight (kg)
2. Read in height (m)
3. Calculate BMI

   BMI = weight/(height * height)
4. Output BMI
5. Output BMI category
6. Prompt user if want to calculate another BMI
7. If yes, go back to step 1
8. If no, end

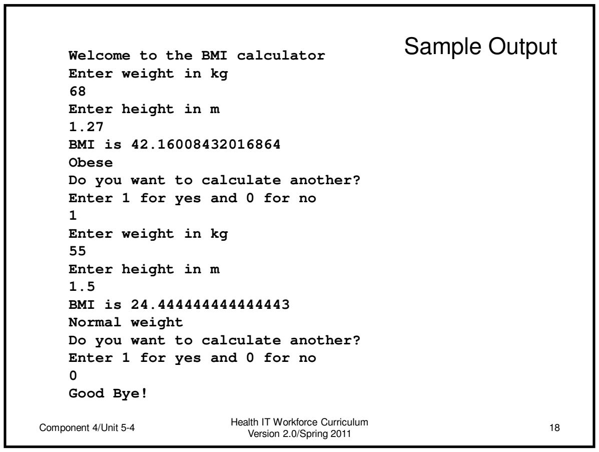
Component 4/Unit 5-4 Health IT Workforce Curriculum Version 2.0/Spring 2011 15



```
import java.util.*;
public class CalcBMI
{
  public static void main(String[] args)
  {
    double bmi, weight, height;
    int anotherBMI;
    Scanner keyboard = new Scanner(System.in);
    System.out.println("Welcome to the BMI calculator");
    anotherBMI = 1;
    while (anotherBMI == 1)
    {
      System.out.println("Enter weight in kg");
      weight = keyboard.nextDouble();
      System.out.println("Enter height in m");
      height = keyboard.nextDouble();
      bmi = weight/(height*height);
      System.out.print("BMI is ");
      System.out.println(bmi);
```

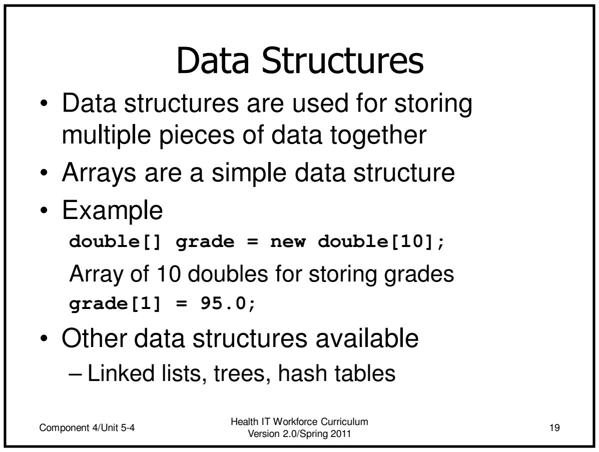


```
   ...
   anotherBMI = 1;
   while (anotherBMI == 1)
   {
       …//input height, weight; calculate BMI
     if (bmi < 18.5)
       System.out.println("Underweight");
     else if ((bmi >= 18.5) && (bmi < 25.0))
       System.out.println("Normal weight");
     else if ((bmi >= 25.0) && (bmi < 30.0))
       System.out.println("Overweight");
     else
       System.out.println("Obese");
     System.out.println("Do you want to calculate another?");
     System.out.println("Enter 1 for yes and 0 for no");
     anotherBMI = keyboard.nextInt();
   }
   System.out.println("Good Bye!");
  }
}
```



Sample Output

```
Welcome to the BMI calculator
Enter weight in kg
68
Enter height in m
1.27
BMI is 42.16008432016864
Obese
Do you want to calculate another?
Enter 1 for yes and 0 for no
1
Enter weight in kg
55
Enter height in m
1.5
BMI is 24.444444444444443
Normal weight
Do you want to calculate another?
Enter 1 for yes and 0 for no
0
Good Bye!
```



Data Structures

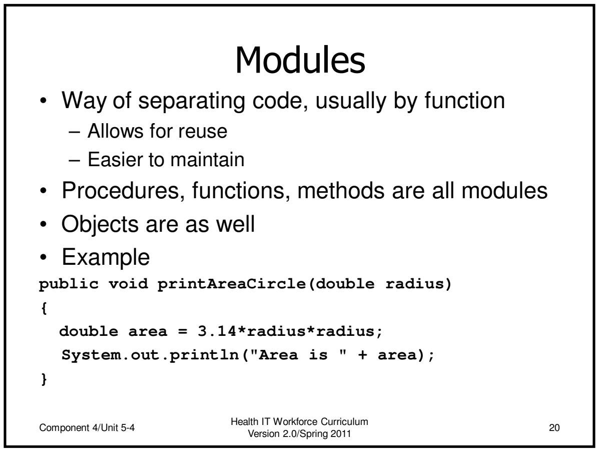
- Data structures are used for storing multiple pieces of data together
- Arrays are a simple data structure
- Example

  ```
  double[] grade = new double[10];
  ```

  Array of 10 doubles for storing grades

  ```
  grade[1] = 95.0;
  ```

- Other data structures available
  - Linked lists, trees, hash tables



Modules

- Way of separating code, usually by function
  - Allows for reuse
  - Easier to maintain
- Procedures, functions, methods are all modules
- Objects are as well
- Example

```
public void printAreaCircle(double radius)
{
  double area = 3.14*radius*radius;
  System.out.println("Area is " + area);
}
```