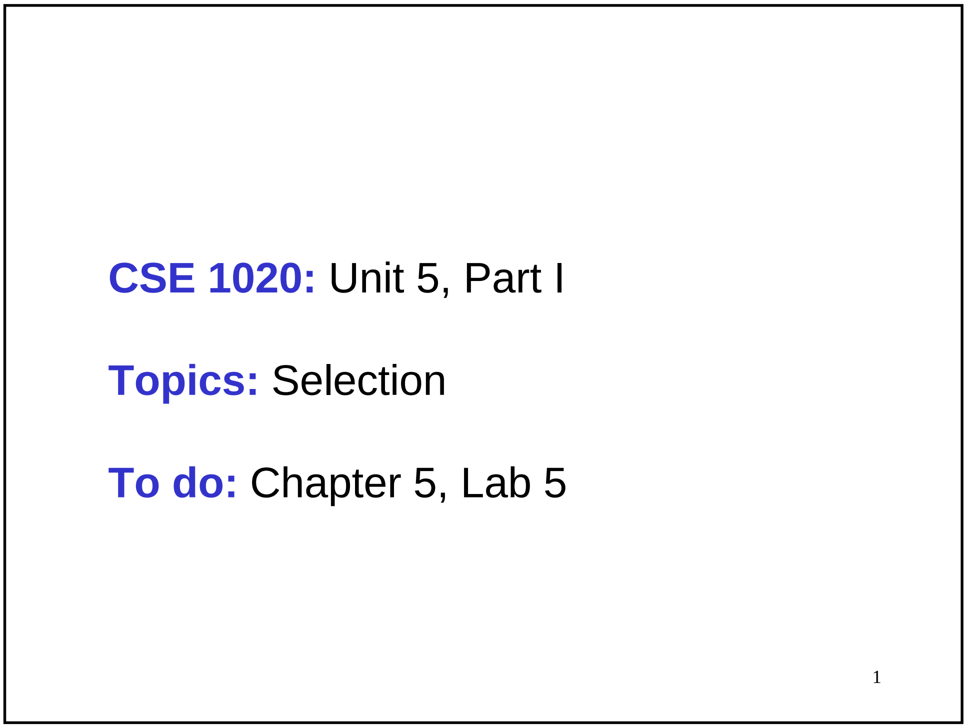
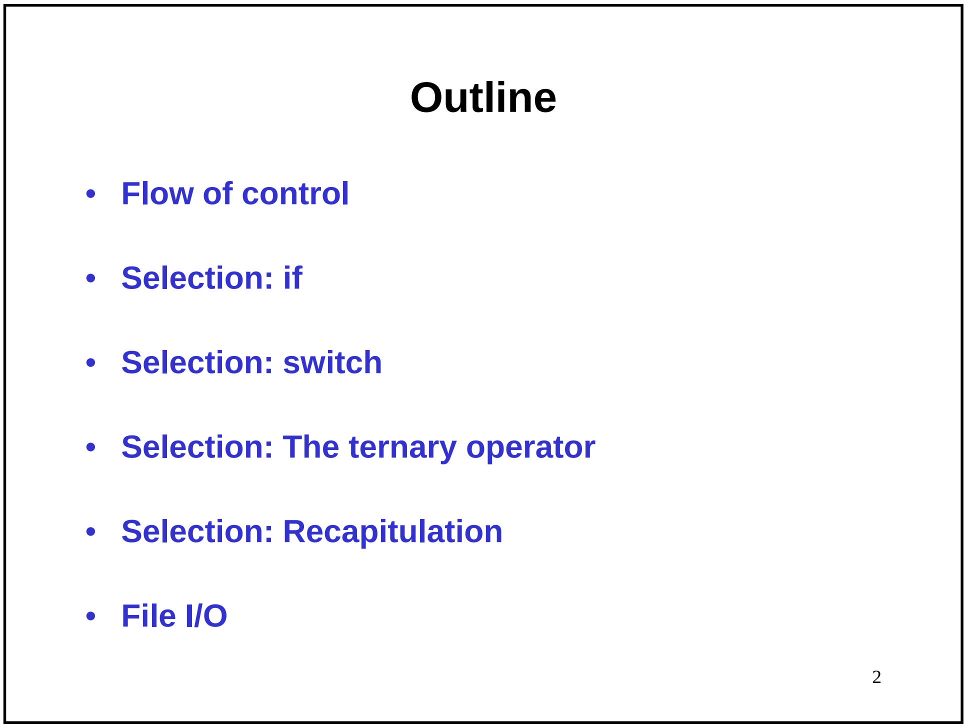


CSE 1020: Unit 5, Part I

Topics: Selection

To do: Chapter 5, Lab 5



Outline

- **Flow of control**
- **Selection: if**
- **Selection: switch**
- **Selection: The ternary operator**
- **Selection: Recapitulation**
- **File I/O**

Outline

- **Flow of control**
- **Selection: if**
- **Selection: switch**
- **Selection: The ternary operator**
- **Selection: Recapitulation**
- **File I/O**

Flow of control

So far…

- The programs we have written have limited flexibility in behaviour.
- Each statement is executed once and in the order that it appears in the program.
- While we have introduced the booleans in support of changing operation based on the truth value of some condition, …
- … we have only used this functionality to abort operation when the condition is violated (e.g., assertions).

Flow of control

Now…

- We introduce two mechanisms in support of increased flexibility of operation.
  1. **Selection:** Take one of several branches depending on a condition.
  2. **Iteration:** Repeat one or more steps depending on a condition.
- Both of these mechanisms employ boolean conditions to make decisions about what actions to perform.

Flow of control

Selection

- We consider two ways to select between alternatives.
  1. Based on the truth value of a single boolean.
  2. Based on scanning against a list of possibilities.

Flow of control

Selection

- We consider two ways to select between alternatives.
  1. Based on the truth value of a single boolean.
  2. Based on scanning against a list of possibilities.

Remark

- We also briefly consider a third, short hand, method.

Outline

- Flow of control
- **Selection: if**
- Selection: switch
- Selection: The ternary operator
- Selection: Recapitulation
- File I/O

Selection: if

If version 1

- The if statement is used to select which is to be performed among some alternative set of operations.
- There are several versions.
- The simplest is used to perform a statement only if a condition (i.e., boolean expression) is true.

Selection: if

If version 1

- The if statement is used to select which is to be performed among some alternative set of operations.
- There are several versions.
- The simplest is used to perform a statement only if a condition (i.e., boolean expression) is true.
- The general form is

```
if ( condition )
   statement
```

- For example

```
double fare = 8.0;
if (age <= 17 )
   fare = 5.0;
```

Selection: if

Nesting of statements

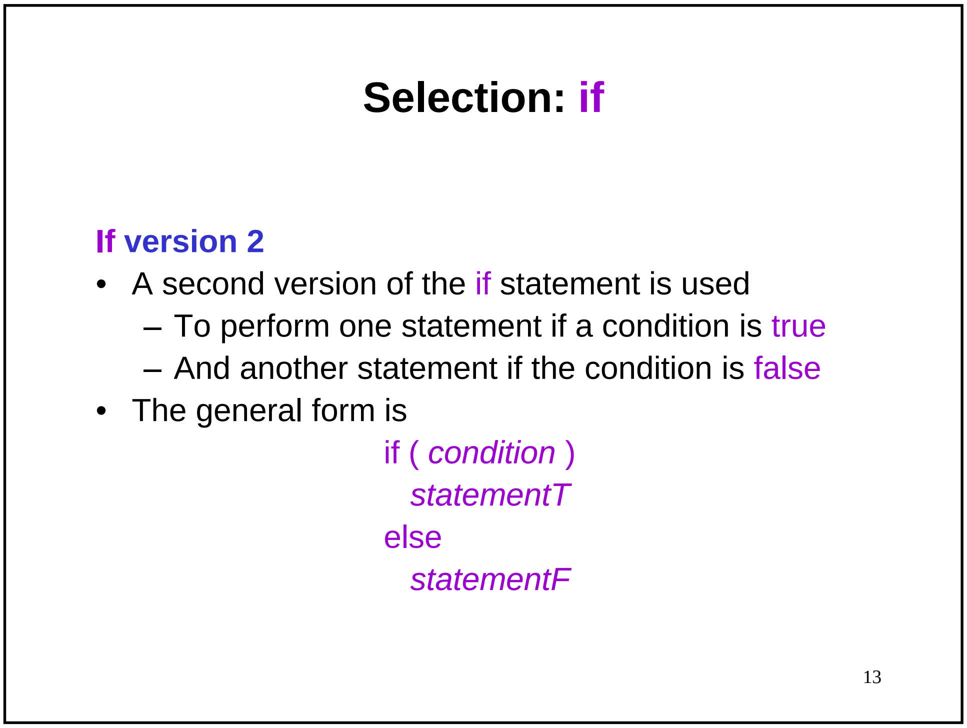
- Sometimes it is desirable to perform several statements if a condition holds.
- We can always group a set of statements by enclosing them between curly brackets {}
  - – We call the enclosed statements a block.

Selection: if

Nesting of statements

- Sometimes it is desirable to perform several statements if a condition holds.
- We can always group a set of statements by enclosing them between curly brackets {}
  - – We call the enclosed statements a block.
- For example

```
output.print(“Enter your age: “);
int age = input.nextInt();
if (age < 0)
{  output.println(“Negative age!”);
   output.print(“Enter your age: “);
   age = input.nextInt();
}
```



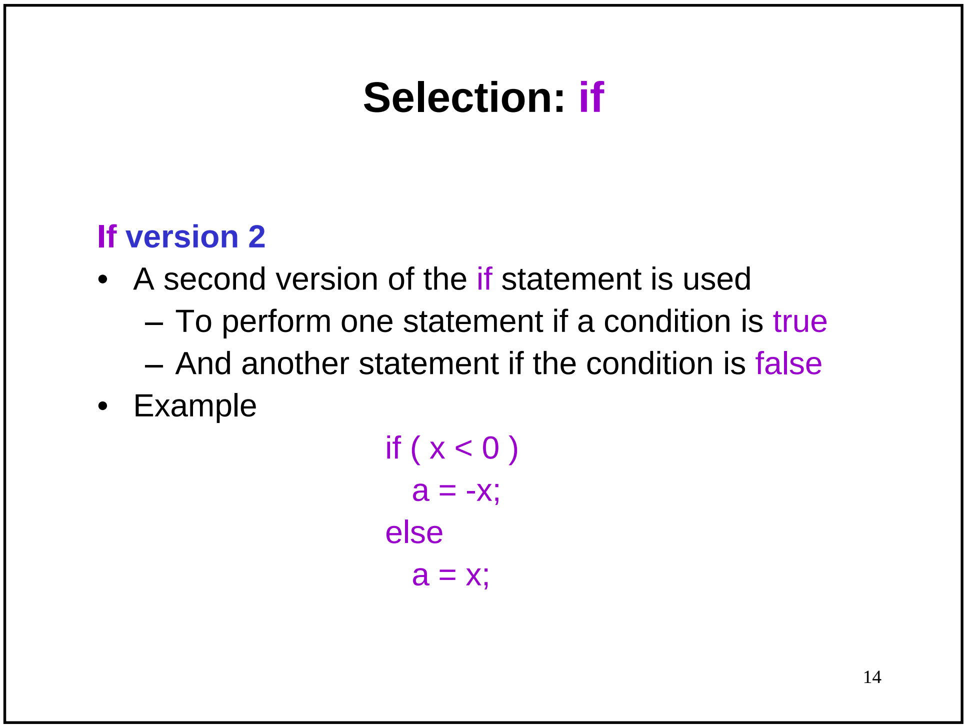
Selection: if

If version 2

- A second version of the if statement is used
  - To perform one statement if a condition is true
  - And another statement if the condition is false
- The general form is

```
if ( condition )
  statementT
else
  statementF
```

13



Selection: if

If version 2

- A second version of the if statement is used
  - To perform one statement if a condition is true
  - And another statement if the condition is false
- Example

```
if ( x < 0 )
  a = -x;
else
  a = x;
```

14

Selection: if

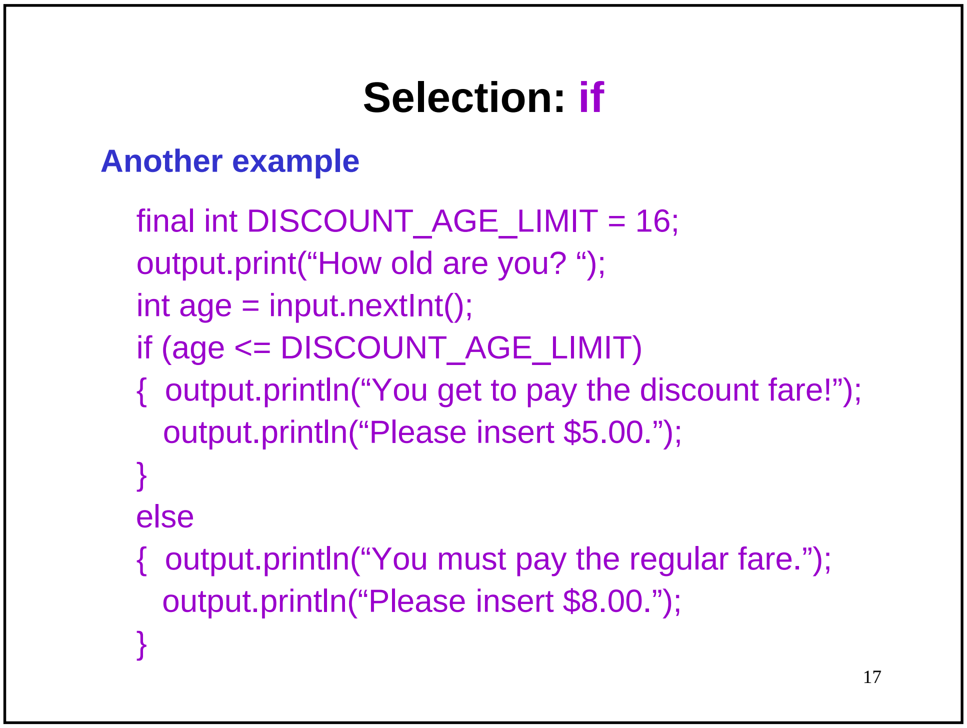
Another example

```
final int DISCOUNT_AGE_LIMIT = 16;
output.print(“How old are you? “);
int age = input.nextInt();
```

Selection: if

Another example

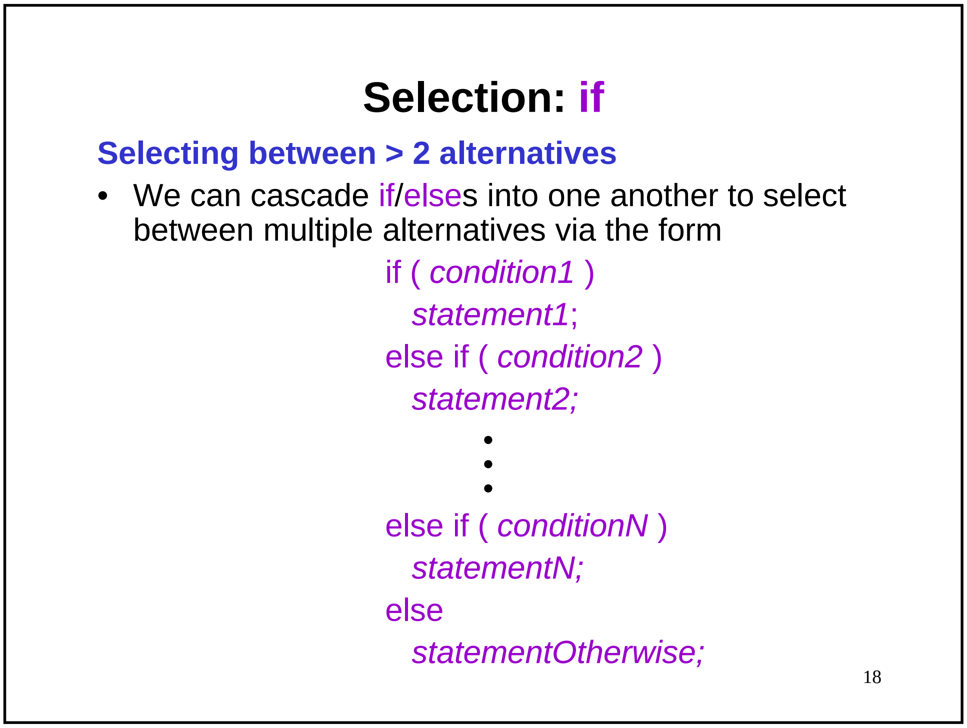
```
final int DISCOUNT_AGE_LIMIT = 16;
output.print(“How old are you? “);
int age = input.nextInt();
if (age <= DISCOUNT_AGE_LIMIT)
{  output.println(“You get to pay the discount fare!”);
   output.println(“Please insert $5.00.”);
}
```



Selection: if

Another example

```
final int DISCOUNT_AGE_LIMIT = 16;
output.print(“How old are you? “);
int age = input.nextInt();
if (age <= DISCOUNT_AGE_LIMIT)
{  output.println(“You get to pay the discount fare!”);
   output.println(“Please insert $5.00.”);
}
else
{  output.println(“You must pay the regular fare.”);
   output.println(“Please insert $8.00.”);
}
```

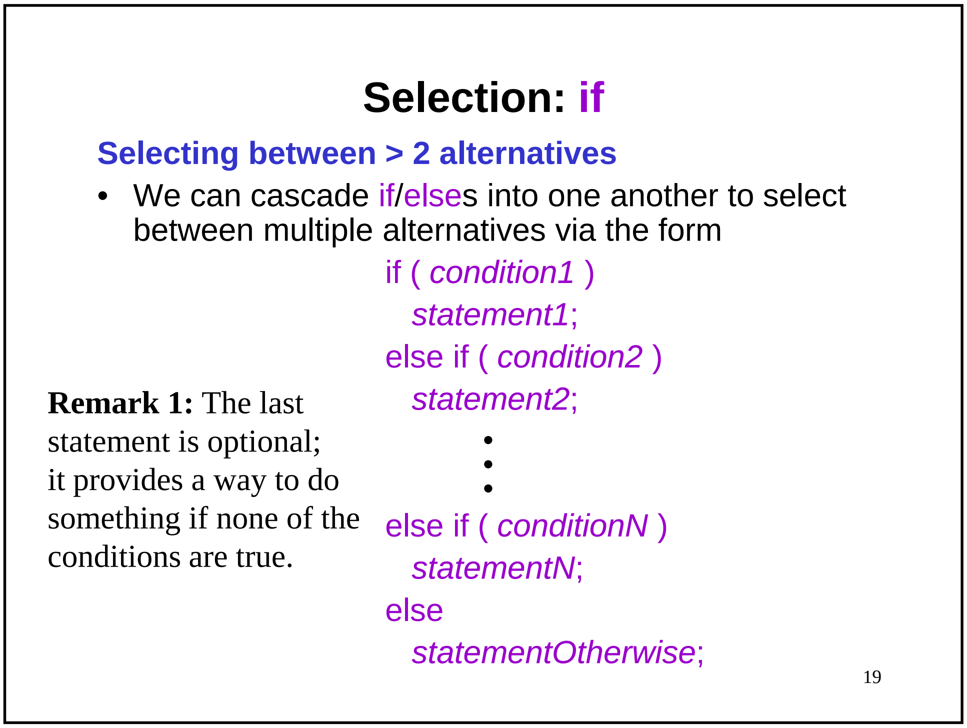


Selection: if

Selecting between > 2 alternatives

- We can cascade if/elses into one another to select between multiple alternatives via the form

```
if ( condition1 )
   statement1;
else if ( condition2 )
   statement2;
       •
       •
       •
else if ( conditionN )
   statementN;
else
   statementOtherwise;
```



Selection: if

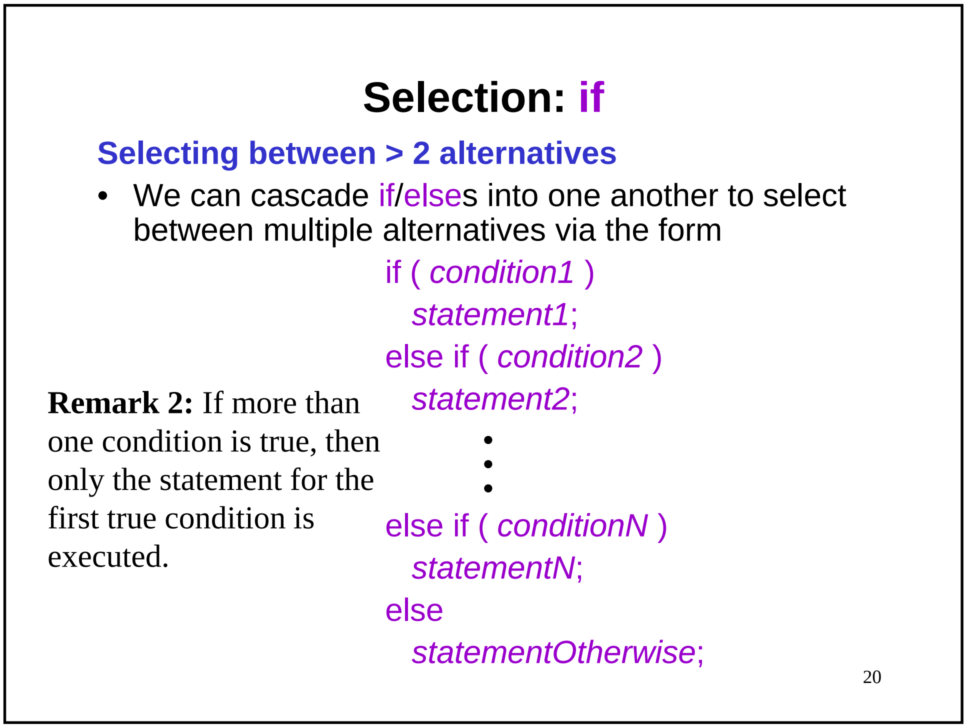
Selecting between > 2 alternatives

- We can cascade if/elses into one another to select between multiple alternatives via the form

```
if ( condition1 )
  statement1;
else if ( condition2 )
  statement2;
        •
        •
        •
else if ( conditionN )
  statementN;
else
  statementOtherwise;
```

Remark 1: The last statement is optional; it provides a way to do something if none of the conditions are true.

19



Selection: if

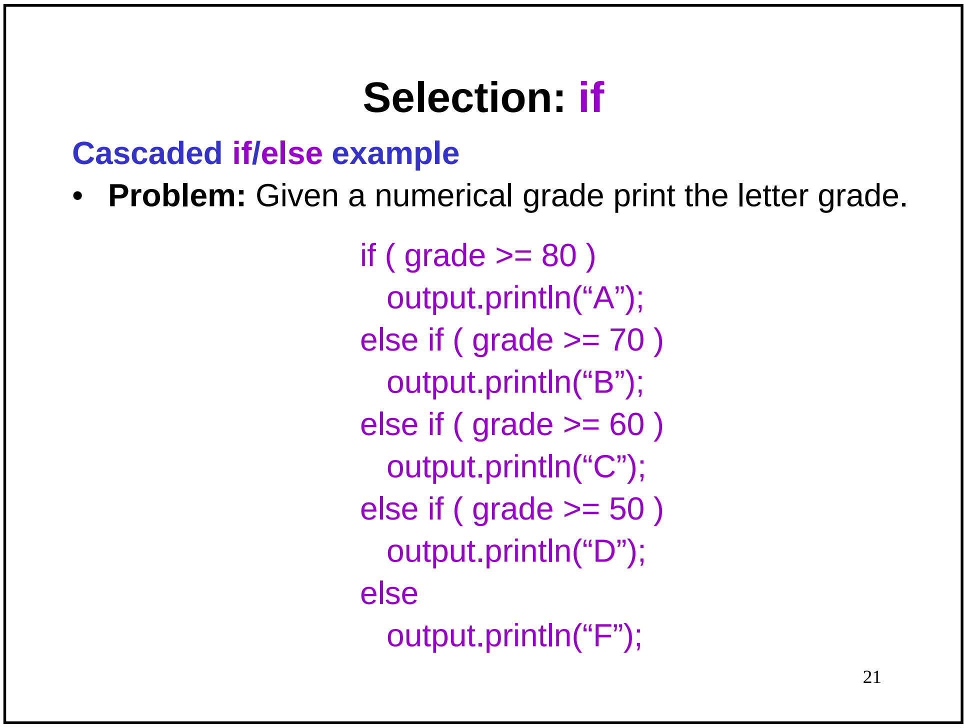
Selecting between > 2 alternatives

- We can cascade if/elses into one another to select between multiple alternatives via the form

```
if ( condition1 )
  statement1;
else if ( condition2 )
  statement2;
        •
        •
        •
else if ( conditionN )
  statementN;
else
  statementOtherwise;
```

Remark 2: If more than one condition is true, then only the statement for the first true condition is executed.

20

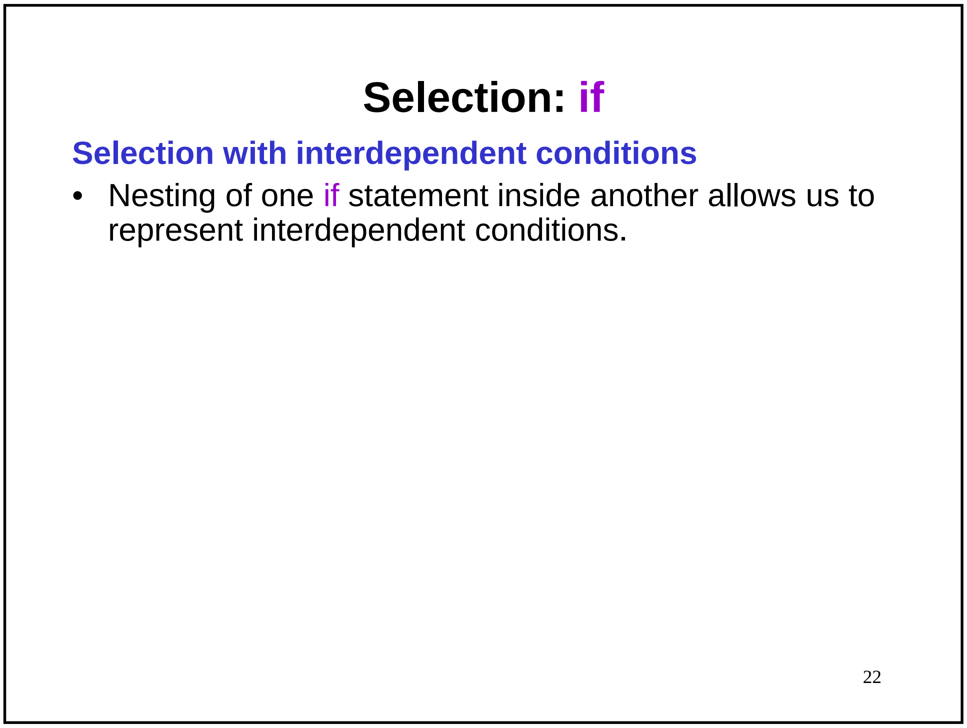


Selection: if

Cascaded if/else example

- **Problem:** Given a numerical grade print the letter grade.

```
if ( grade >= 80 )
  output.println(“A”);
else if ( grade >= 70 )
  output.println(“B”);
else if ( grade >= 60 )
  output.println(“C”);
else if ( grade >= 50 )
  output.println(“D”);
else
  output.println(“F”);
```



Selection: if

Selection with interdependent conditions

- Nesting of one if statement inside another allows us to represent interdependent conditions.

Selection: if

Selection with interdependent conditions

```
final int HEADS = 1;
final int RECEIVE = 1;
output.print(“Enter 1 for heads and 2 for tails: “);
int coin = input.nextInt();
output.print(“Enter 1 to receive and 2 to kickoff: “);
int choice = input.nextInt();
```

Selection: if

Selection with interdependent conditions

```
final int HEADS = 1;
final int RECEIVE = 1;
output.print(“Enter 1 for heads and 2 for tails: “);
int coin = input.nextInt();
output.print(“Enter 1 to receive and 2 to kickoff: “);
int choice = input.nextInt();
if (coin == HEADS )
   if (choice == RECEIVE);
      output.println(“You won the toss and will receive.”);
```

Selection: if

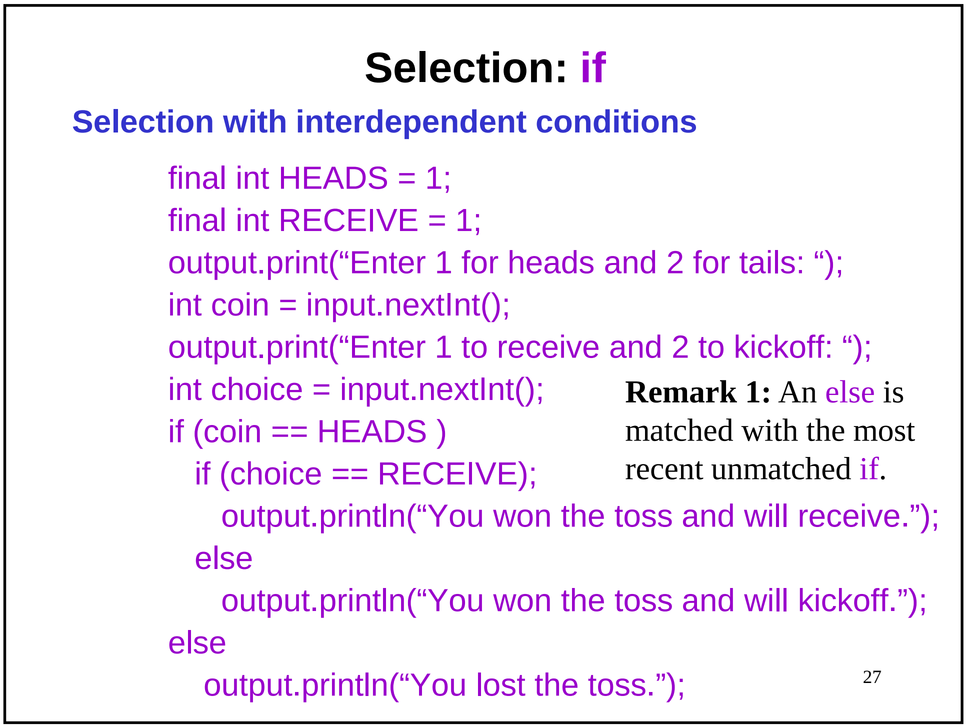
Selection with interdependent conditions

```
final int HEADS = 1;
final int RECEIVE = 1;
output.print(“Enter 1 for heads and 2 for tails: “);
int coin = input.nextInt();
output.print(“Enter 1 to receive and 2 to kickoff: “);
int choice = input.nextInt();
if (coin == HEADS )
  if (choice == RECEIVE);
    output.println(“You won the toss and will receive.”);
  else
    output.println(“You won the toss and will kickoff.”);
```

Selection: if

Selection with interdependent conditions

```
final int HEADS = 1;
final int RECEIVE = 1;
output.print(“Enter 1 for heads and 2 for tails: “);
int coin = input.nextInt();
output.print(“Enter 1 to receive and 2 to kickoff: “);
int choice = input.nextInt();
if (coin == HEADS )
  if (choice == RECEIVE);
    output.println(“You won the toss and will receive.”);
  else
    output.println(“You won the toss and will kickoff.”);
else
   output.println(“You lost the toss.”);
```

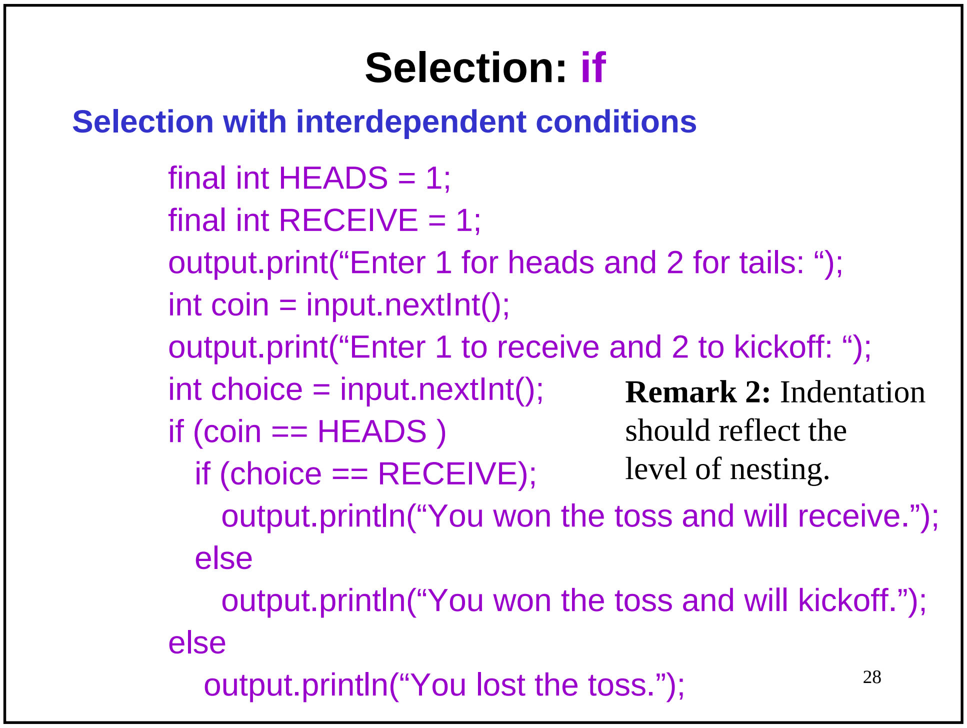


Selection: if

Selection with interdependent conditions

```
final int HEADS = 1;
final int RECEIVE = 1;
output.print(“Enter 1 for heads and 2 for tails: “);
int coin = input.nextInt();
output.print(“Enter 1 to receive and 2 to kickoff: “);
int choice = input.nextInt();
if (coin == HEADS )
  if (choice == RECEIVE);
    output.println(“You won the toss and will receive.”);
  else
    output.println(“You won the toss and will kickoff.”);
else
   output.println(“You lost the toss.”);
```

Remark 1: An else is matched with the most recent unmatched if.



Selection: if

Selection with interdependent conditions

```
final int HEADS = 1;
final int RECEIVE = 1;
output.print(“Enter 1 for heads and 2 for tails: “);
int coin = input.nextInt();
output.print(“Enter 1 to receive and 2 to kickoff: “);
int choice = input.nextInt();
if (coin == HEADS )
  if (choice == RECEIVE);
    output.println(“You won the toss and will receive.”);
  else
    output.println(“You won the toss and will kickoff.”);
else
   output.println(“You lost the toss.”);
```

Remark 2: Indentation should reflect the level of nesting.

Selection: if

Exercise

- For each possible value of cond1 and cond2, state the value of r after the code is executed.

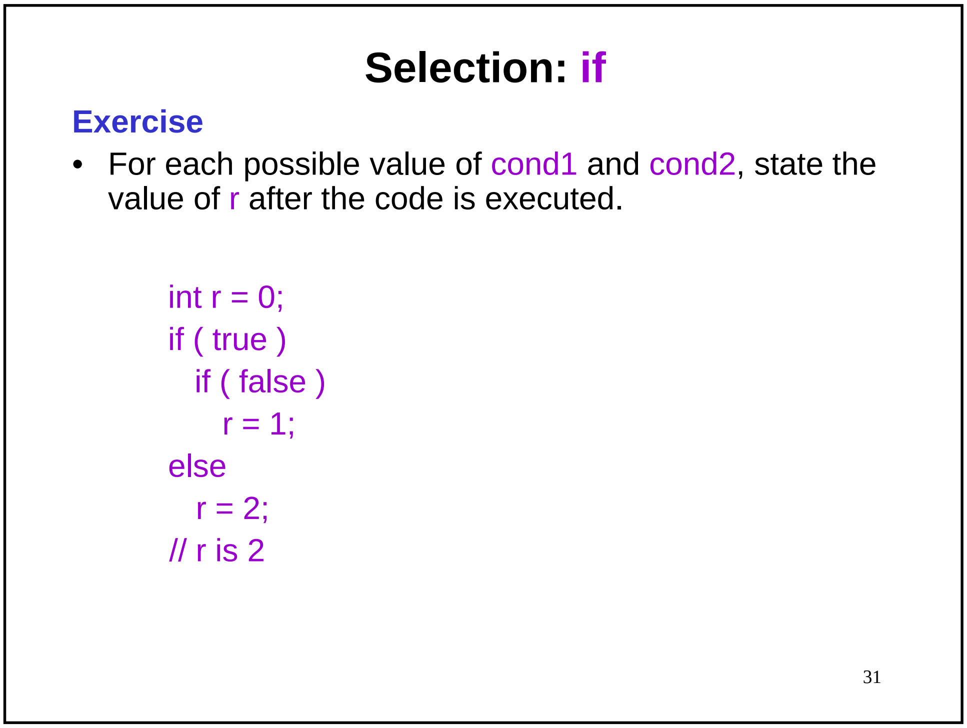
```
int r = 0;
if ( cond1 )
   if ( cond2 )
      r = 1;
else
   r = 2;
```

Selection: if

Exercise

- For each possible value of cond1 and cond2, state the value of r after the code is executed.

```
int r = 0;
if ( true )
   if ( true )
      r = 1;
else
   r = 2;
// r is 1
```

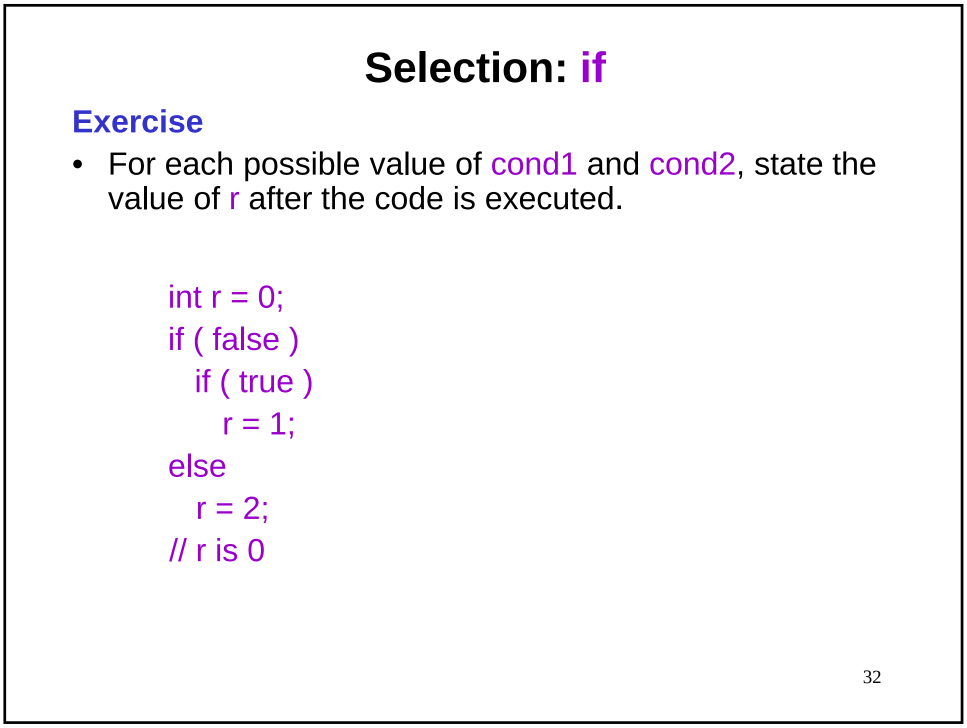


Selection: if

Exercise

- For each possible value of cond1 and cond2, state the value of r after the code is executed.

```
int r = 0;
if ( true )
   if ( false )
      r = 1;
else
   r = 2;
// r is 2
```

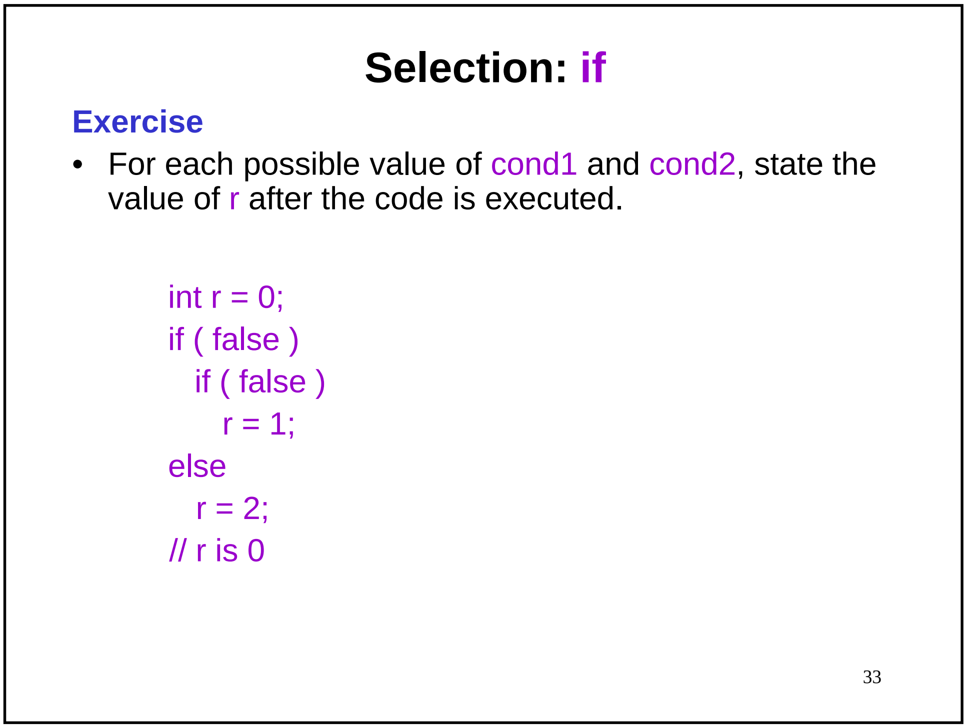


Selection: if

Exercise

- For each possible value of cond1 and cond2, state the value of r after the code is executed.

```
int r = 0;
if ( false )
   if ( true )
      r = 1;
else
   r = 2;
// r is 0
```

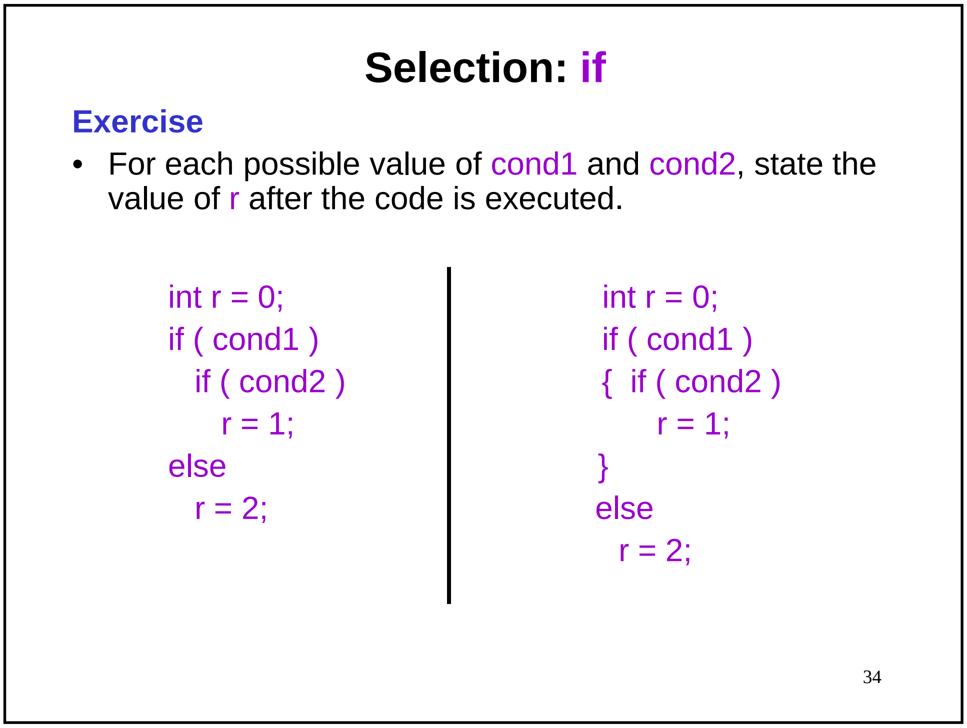


Selection: if

Exercise

- For each possible value of cond1 and cond2, state the value of r after the code is executed.

```
int r = 0;
if ( false )
   if ( false )
      r = 1;
else
   r = 2;
// r is 0
```



Selection: if

Exercise

- For each possible value of cond1 and cond2, state the value of r after the code is executed.

```
int r = 0;
if ( cond1 )
   if ( cond2 )
      r = 1;
else
   r = 2;
```

```
int r = 0;
if ( cond1 )
{  if ( cond2 )
      r = 1;
}
else
   r = 2;
```

Selection: if

Exercise

- For each possible value of cond1 and cond2, state the value of r after the code is executed.

```
int r = 0;
if ( true )
   if ( true )
      r = 1;
else
   r = 2;
// r is 1
```

```
int r = 0;
if ( true )
{  if ( true )
      r = 1;
}
else
   r = 2;
// r is 1
```

Selection: if

Exercise

- For each possible value of cond1 and cond2, state the value of r after the code is executed.

```
int r = 0;
if ( true )
   if ( false )
      r = 1;
else
   r = 2;
// r is 2
```

```
int r = 0;
if ( true )
{  if ( false )
      r = 1;
}
else
   r = 2;
// r is 0
```

Selection: if

Exercise

- For each possible value of cond1 and cond2, state the value of r after the code is executed.

```
int r = 0;
if ( false )
   if ( true )
      r = 1;
else
   r = 2;
// r is 0
```

```
int r = 0;
if ( false )
{  if ( true )
      r = 1;
}
else
  r = 2;
// r is 2
```

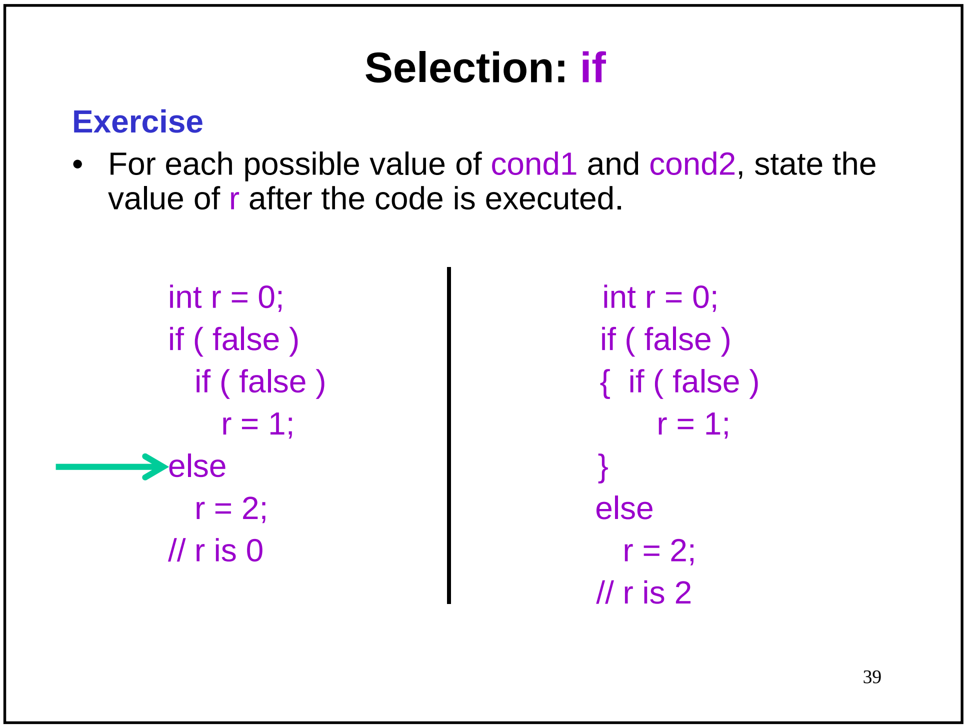
Selection: if

Exercise

- For each possible value of cond1 and cond2, state the value of r after the code is executed.

```
int r = 0;
if ( false )
   if ( false )
      r = 1;
else
   r = 2;
// r is 0
```

```
int r = 0;
if ( false )
{  if ( false )
      r = 1;
}
else
   r = 2;
// r is 2
```



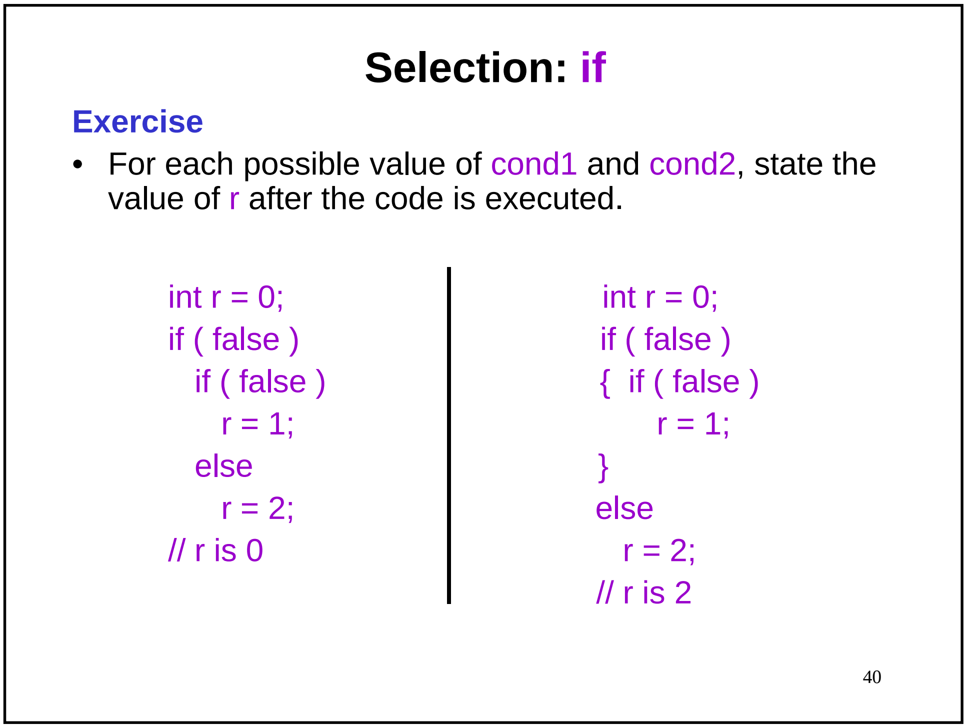
Selection: if

Exercise

- For each possible value of cond1 and cond2, state the value of r after the code is executed.

```
int r = 0;
if ( false )
   if ( false )
      r = 1;
→else
   r = 2;
// r is 0
```

```
int r = 0;
if ( false )
{  if ( false )
      r = 1;
}
else
   r = 2;
// r is 2
```



Selection: if

Exercise

- For each possible value of cond1 and cond2, state the value of r after the code is executed.

```
int r = 0;
if ( false )
   if ( false )
      r = 1;
   else
      r = 2;
// r is 0
```

```
int r = 0;
if ( false )
{  if ( false )
      r = 1;
}
else
   r = 2;
// r is 2
```

Selection: if

MkChange revisited

- Recall the MkChange software that we developed.
- For an input amount of money (CND) in cents it returned the change in quarters, dimes, nickels and pennies.
- For example

```
% java MkChange
```

Selection: if

MkChange revisited

- Recall the MkChange software that we developed.
- For an input amount of money (CND) in cents it returned the change in quarters, dimes, nickels and pennies.
- For example

```
% java MkChange
Enter the amount in cents:
```

Selection: if

MkChange revisited

- Recall the MkChange software that we developed.
- For an input amount of money (CND) in cents it returned the change in quarters, dimes, nickels and pennies.
- For example

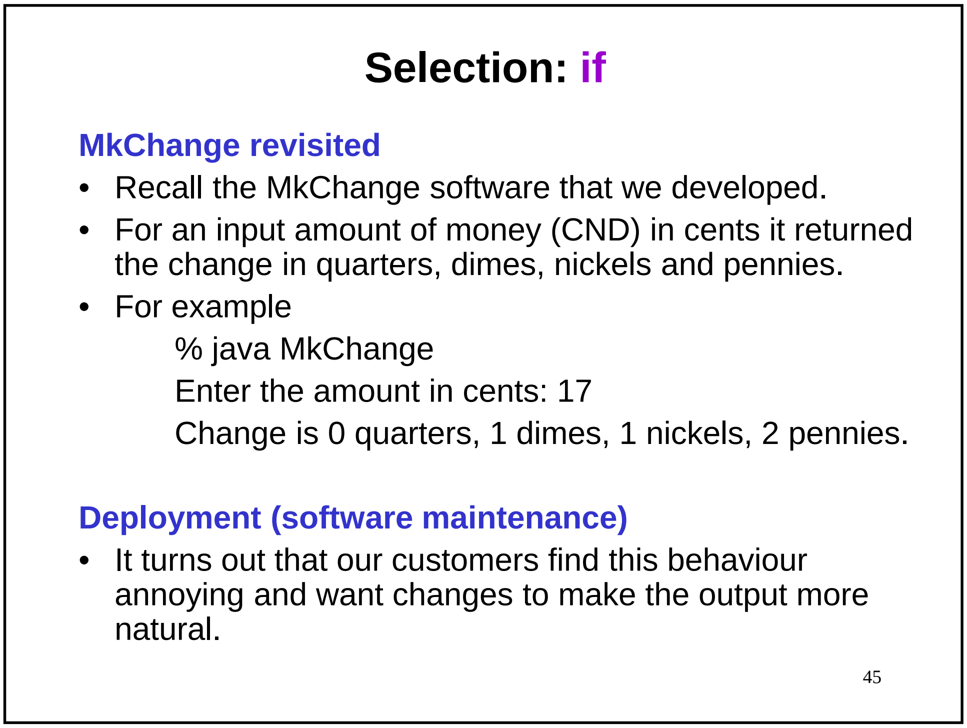
```
% java MkChange
Enter the amount in cents: 17
```

Selection: if

MkChange revisited

- Recall the MkChange software that we developed.
- For an input amount of money (CND) in cents it returned the change in quarters, dimes, nickels and pennies.
- For example

```
% java MkChange
Enter the amount in cents: 17
Change is 0 quarters, 1 dimes, 1 nickels, 2 pennies.
```



Selection: if

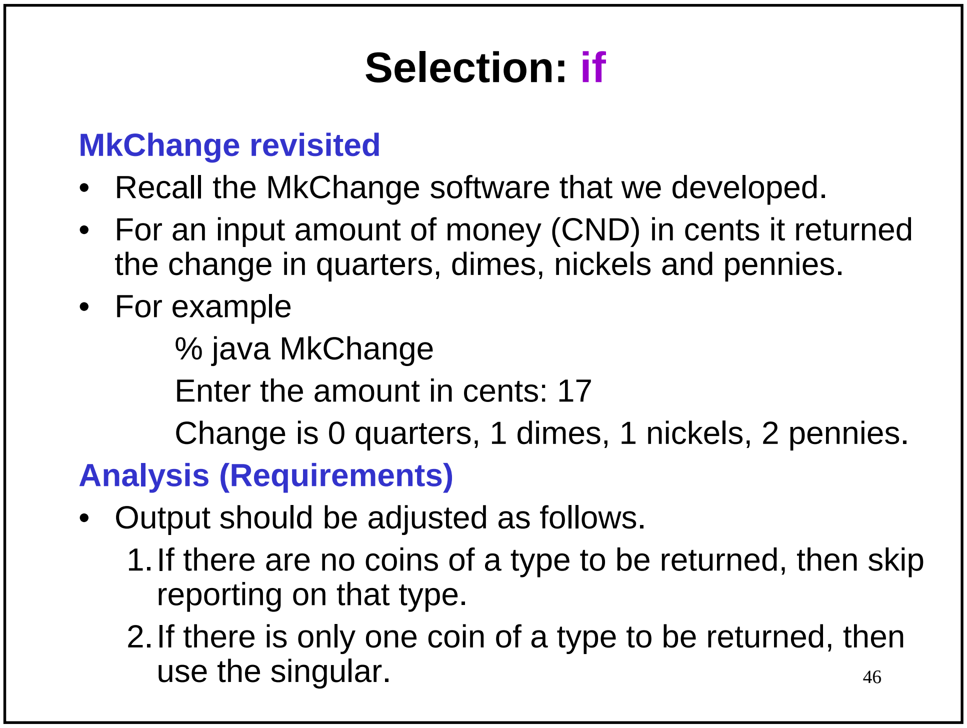
MkChange revisited

- Recall the MkChange software that we developed.
- For an input amount of money (CND) in cents it returned the change in quarters, dimes, nickels and pennies.
- For example

```
% java MkChange
Enter the amount in cents: 17
Change is 0 quarters, 1 dimes, 1 nickels, 2 pennies.
```

Deployment (software maintenance)

- It turns out that our customers find this behaviour annoying and want changes to make the output more natural.



Selection: if

MkChange revisited

- Recall the MkChange software that we developed.
- For an input amount of money (CND) in cents it returned the change in quarters, dimes, nickels and pennies.
- For example

```
% java MkChange
Enter the amount in cents: 17
Change is 0 quarters, 1 dimes, 1 nickels, 2 pennies.
```

Analysis (Requirements)

- Output should be adjusted as follows.
  1. If there are no coins of a type to be returned, then skip reporting on that type.
  2. If there is only one coin of a type to be returned, then use the singular.

Selection: if

Design

- We can make the desired changes by using if statements at time of output.

```
if the number of coins > 1
        report using “coins”
else if number of coins == 1
        report using “coin”
else skip this coin
```

- We repeat this construction for coin ∈{quarters, dimes, nickels, pennies}.

Selection: if

Implementation

```
// Output
output.print(“Change is ”);
output.print(nQuarters + “ quarters, ”);
```

Selection: if

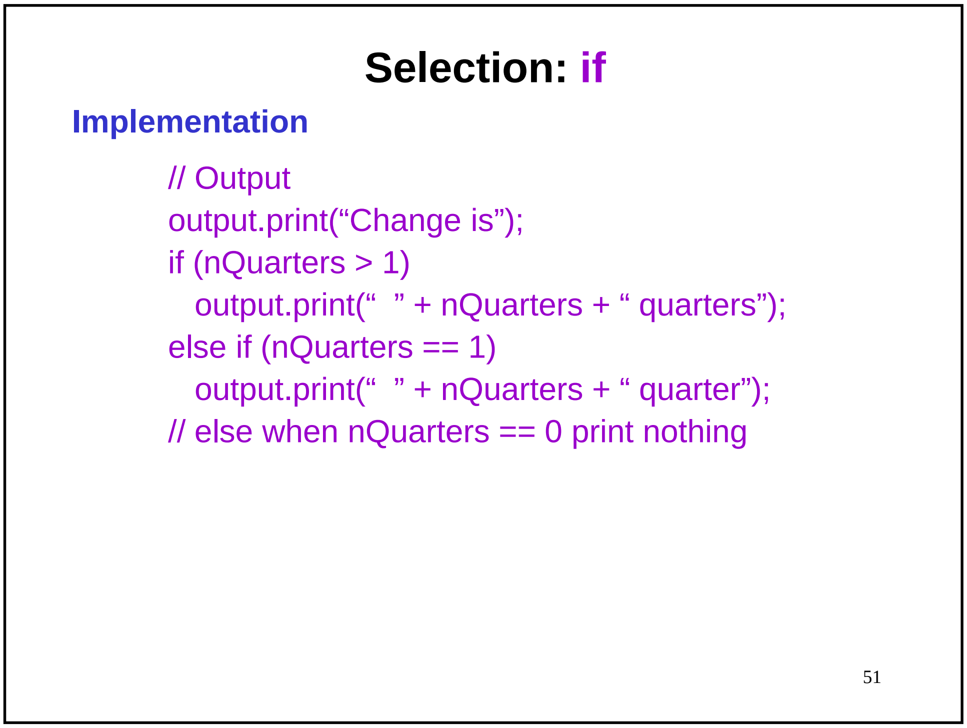
Implementation

```
// Output
IO.print(“Change is ”);
~~IO.print(nQuarters + “ quarters ”);~~
```

Selection: if

Implementation

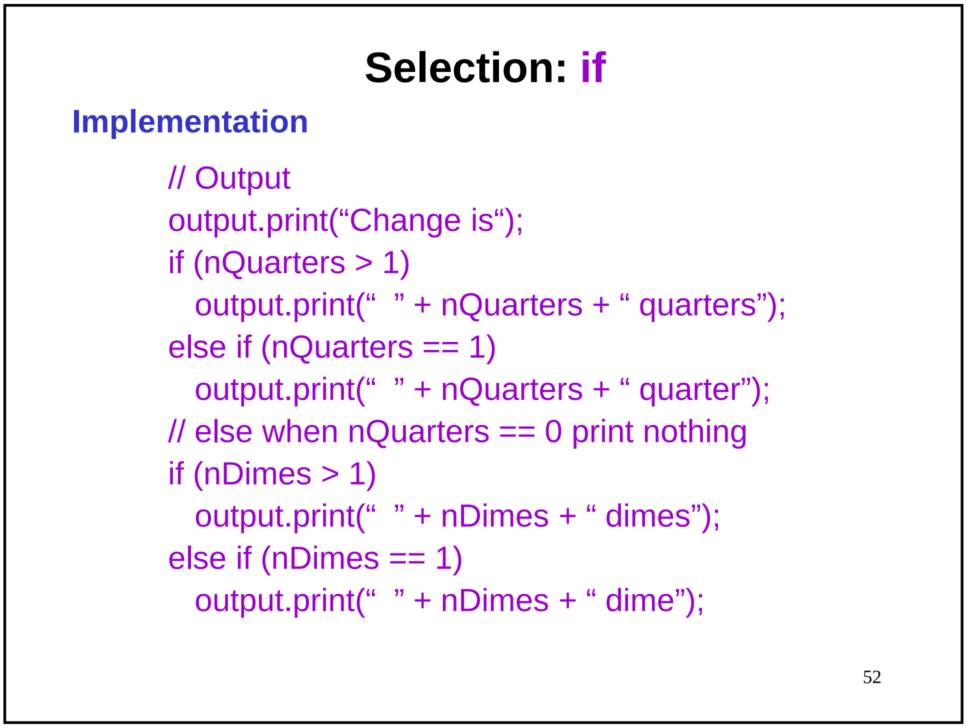
```
// Output
IO.print(“Change is ”);
if the number of coins > 1
        report using “coins”
else if number of coins == 1
        report using “coin”
else skip this coin
```



Selection: if

Implementation

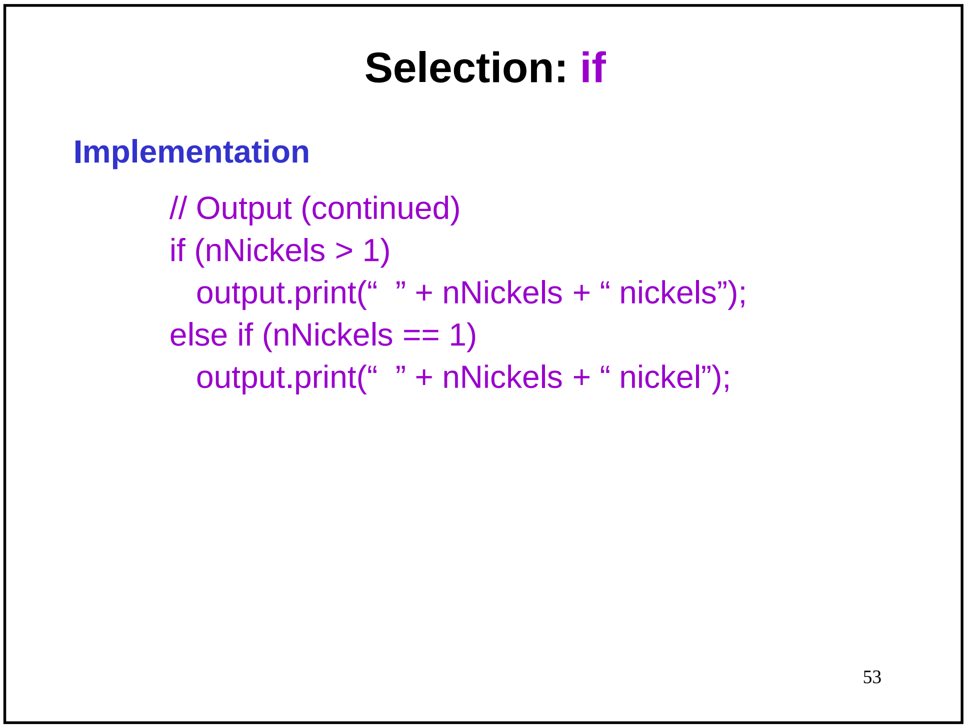
```
// Output
output.print(“Change is”);
if (nQuarters > 1)
  output.print(“  ” + nQuarters + “ quarters”);
else if (nQuarters == 1)
  output.print(“  ” + nQuarters + “ quarter”);
// else when nQuarters == 0 print nothing
```



Selection: if

Implementation

```
// Output
output.print(“Change is“);
if (nQuarters > 1)
  output.print(“  ” + nQuarters + “ quarters”);
else if (nQuarters == 1)
  output.print(“  ” + nQuarters + “ quarter”);
// else when nQuarters == 0 print nothing
if (nDimes > 1)
  output.print(“  ” + nDimes + “ dimes”);
else if (nDimes == 1)
  output.print(“  ” + nDimes + “ dime”);
```

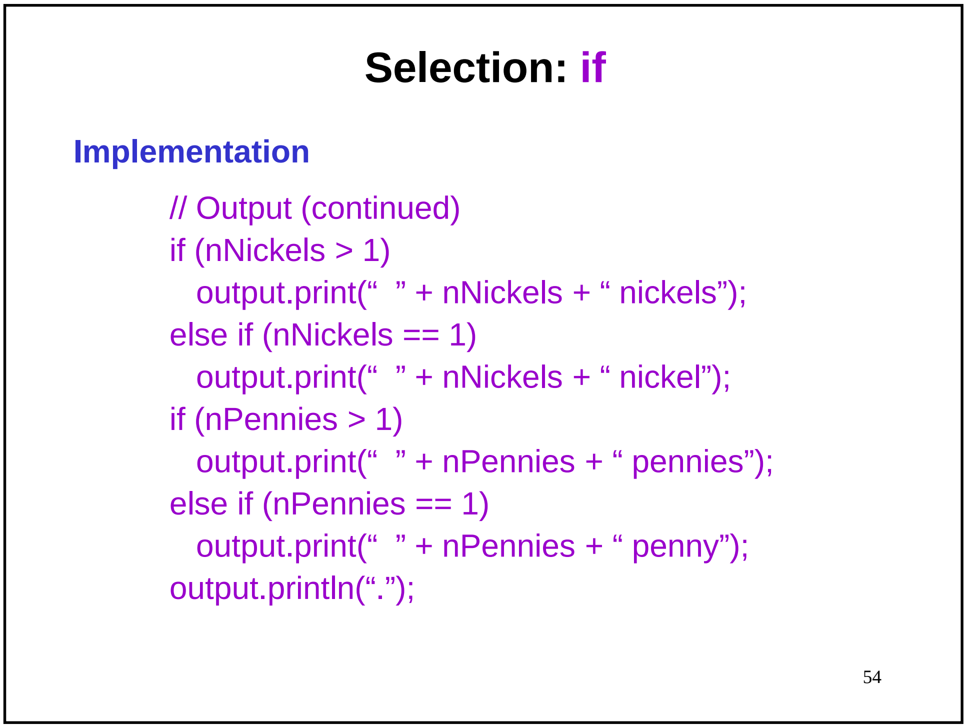


Selection: if

Implementation

```
// Output (continued)
if (nNickels > 1)
   output.print("  " + nNickels + " nickels");
else if (nNickels == 1)
   output.print("  " + nNickels + " nickel");
```

53



Selection: if

Implementation

```
// Output (continued)
if (nNickels > 1)
   output.print("  " + nNickels + " nickels");
else if (nNickels == 1)
   output.print("  " + nNickels + " nickel");
if (nPennies > 1)
   output.print("  " + nPennies + " pennies");
else if (nPennies == 1)
   output.print("  " + nPennies + " penny");
output.println(".");
```

54

Selection: if

Testing

- We continue in the edit/compile/run loop until we have a nominally working MkChange3.class
- As a test of our modified code, let’s try out the offending case that was presented earlier

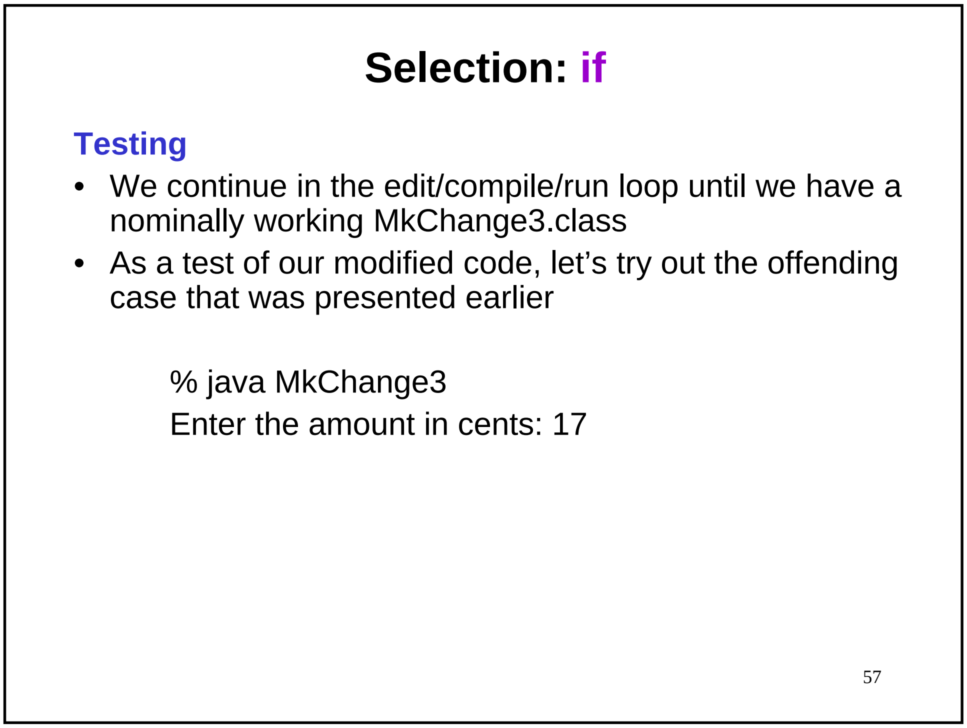
```
% java MkChange3
```

Selection: if

Testing

- We continue in the edit/compile/run loop until we have a nominally working MkChange3.class
- As a test of our modified code, let’s try out the offending case that was presented earlier

```
% java MkChange3
Enter the amount in cents:
```



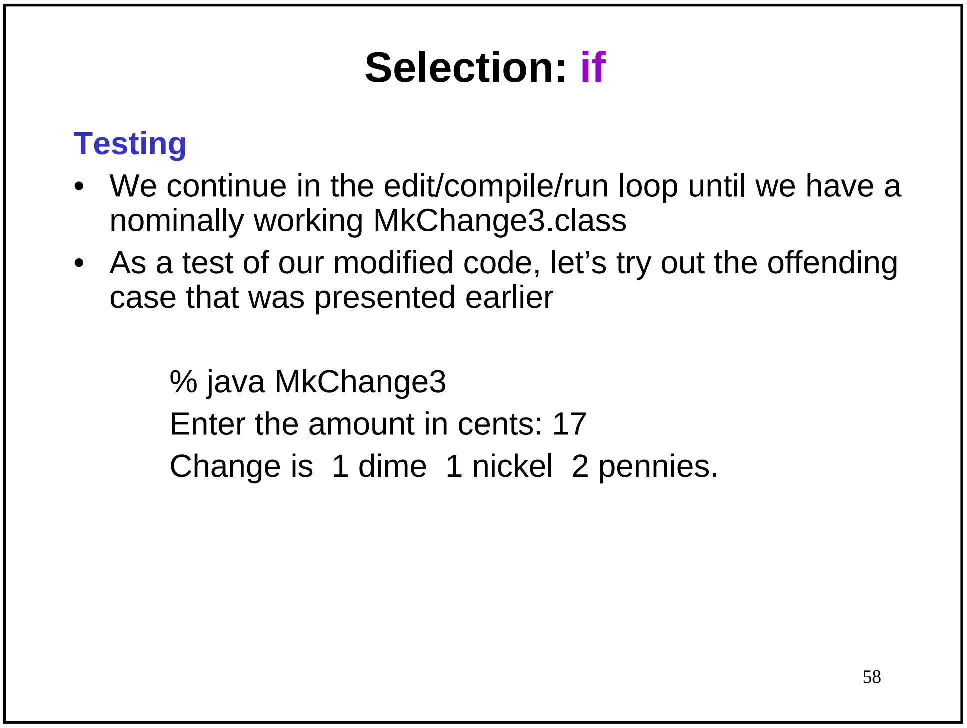
Selection: if

Testing

- We continue in the edit/compile/run loop until we have a nominally working MkChange3.class
- As a test of our modified code, let’s try out the offending case that was presented earlier

```
% java MkChange3
Enter the amount in cents: 17
```

57



Selection: if

Testing

- We continue in the edit/compile/run loop until we have a nominally working MkChange3.class
- As a test of our modified code, let’s try out the offending case that was presented earlier

```
% java MkChange3
Enter the amount in cents: 17
Change is  1 dime  1 nickel  2 pennies.
```

58

Selection: if

Testing

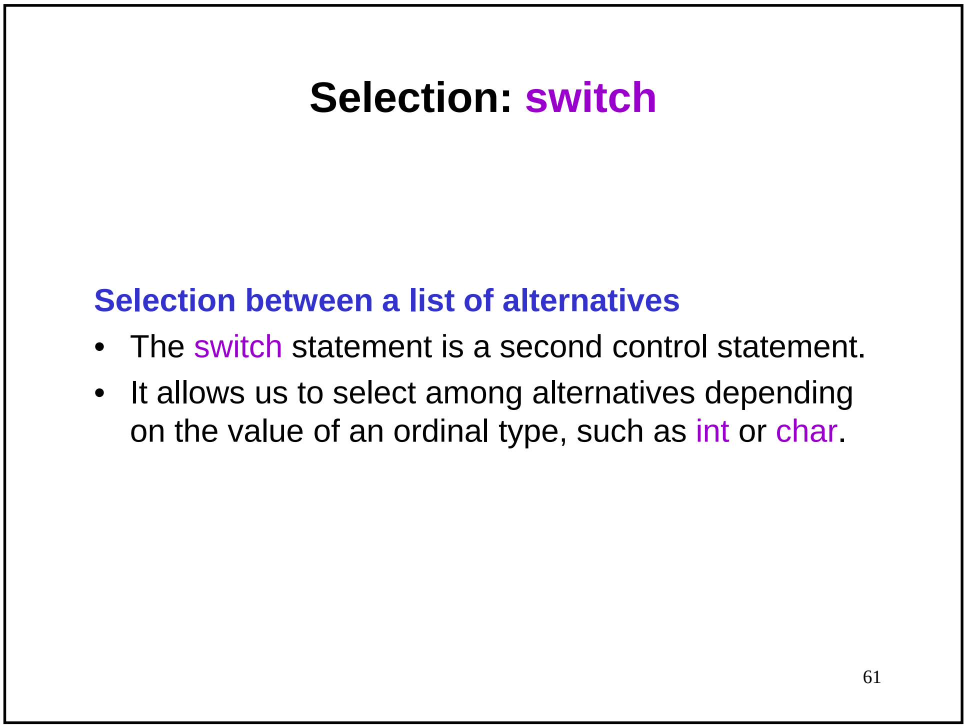
- We continue in the edit/compile/run loop until we have a nominally working MkChange3.class
- As a test of our modified code, let’s try out the offending case that was presented earlier

```
% java MkChange3
Enter the amount in cents: 17
Change is  1 dime  1 nickel  2 pennies.
```

- In practice, much more extensive testing would be appropriate.

Outline

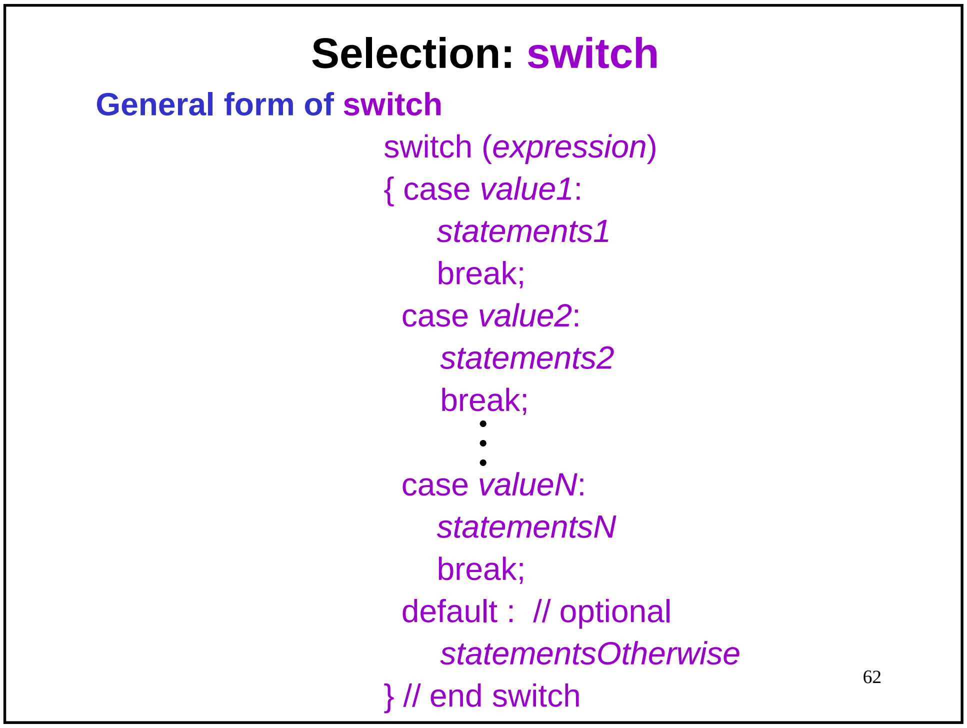
- Flow of control
- Selection: if
- **Selection: switch**
- Selection: The ternary operator
- Selection: Recapitulation
- File I/O



Selection: switch

Selection between a list of alternatives

- The switch statement is a second control statement.
- It allows us to select among alternatives depending on the value of an ordinal type, such as int or char.



Selection: switch

General form of switch

```
switch (expression)
{ case value1:
      statements1
      break;
  case value2:
      statements2
      break;
        •
        •
        •
  case valueN:
      statementsN
      break;
  default :  // optional
      statementsOtherwise
} // end switch
```

Selection: switch

General form of switch

```
switch (expression)
{ case value1:
     statements1
     break;
  case value2:
     statements2
     break;
        :
  case valueN:
     statementsN
     break;
  default :  // optional
     statementsOtherwise
} // end switch
```

Remark 1: statementsK are executed when expression has valueK.

Selection: switch

General form of switch

```
switch (expression)
{ case value1:
     statements1
     break;
  case value2:
     statements2
     break;
        :
  case valueN:
     statementsN
     break;
  default :  // optional
     statementsOtherwise
} // end switch
```

Remark 2: statementsOtherwise are executed when expression has a value different from all the cases.

Selection: switch

General form of switch

```
switch (expression)
{ case value1:
     statements1
     break;
  case value2:
     statements2
     break;
       .
       .
       .
  case valueN:
     statementsN
     break;
  default :  // optional
     statementsOtherwise
} // end switch
```

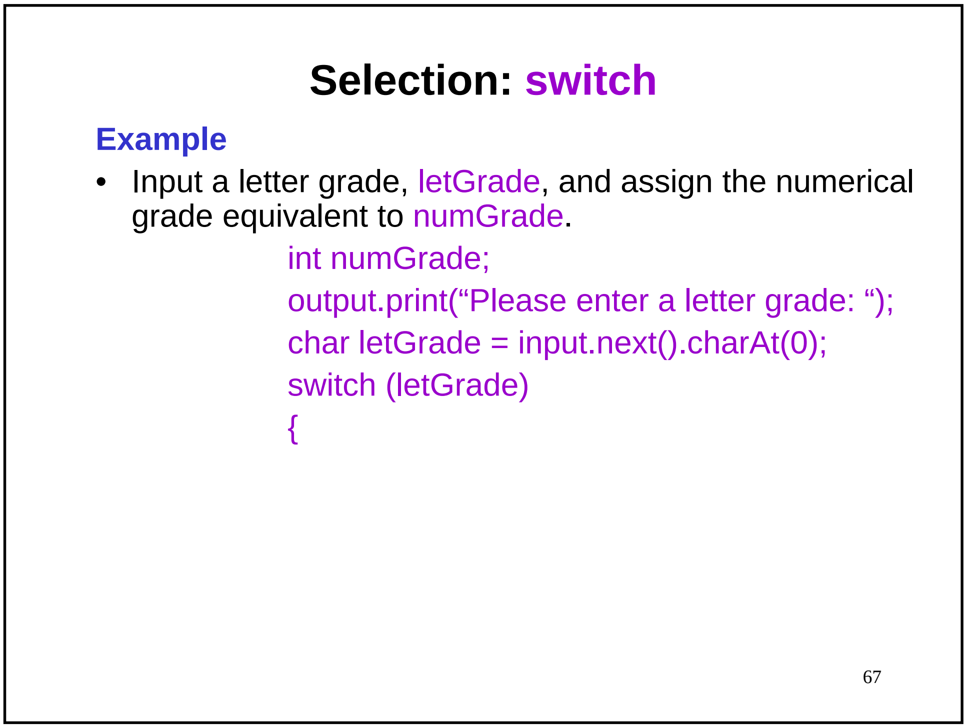
Remark 3: break sends execution of the program beyond the switch so as to resume at the statement following }.

Selection: switch

Example

- Input a letter grade, letGrade, and assign the numerical grade equivalent to numGrade.

```
int numGrade;
output.print(“Please enter a letter grade: “);
char letGrade = input.next().charAt(0);
```



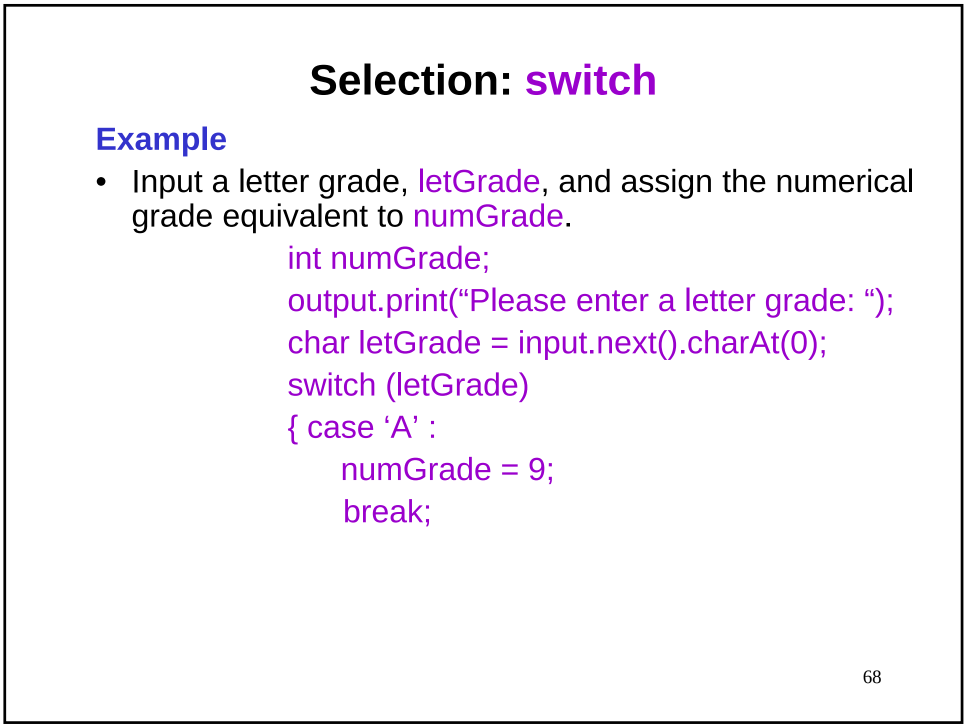
Selection: switch

Example

- Input a letter grade, letGrade, and assign the numerical grade equivalent to numGrade.

```
int numGrade;
output.print(“Please enter a letter grade: “);
char letGrade = input.next().charAt(0);
switch (letGrade)
{
```

67



Selection: switch

Example

- Input a letter grade, letGrade, and assign the numerical grade equivalent to numGrade.

```
int numGrade;
output.print(“Please enter a letter grade: “);
char letGrade = input.next().charAt(0);
switch (letGrade)
{ case ‘A’ :
      numGrade = 9;
      break;
```

68

Selection: switch

Example

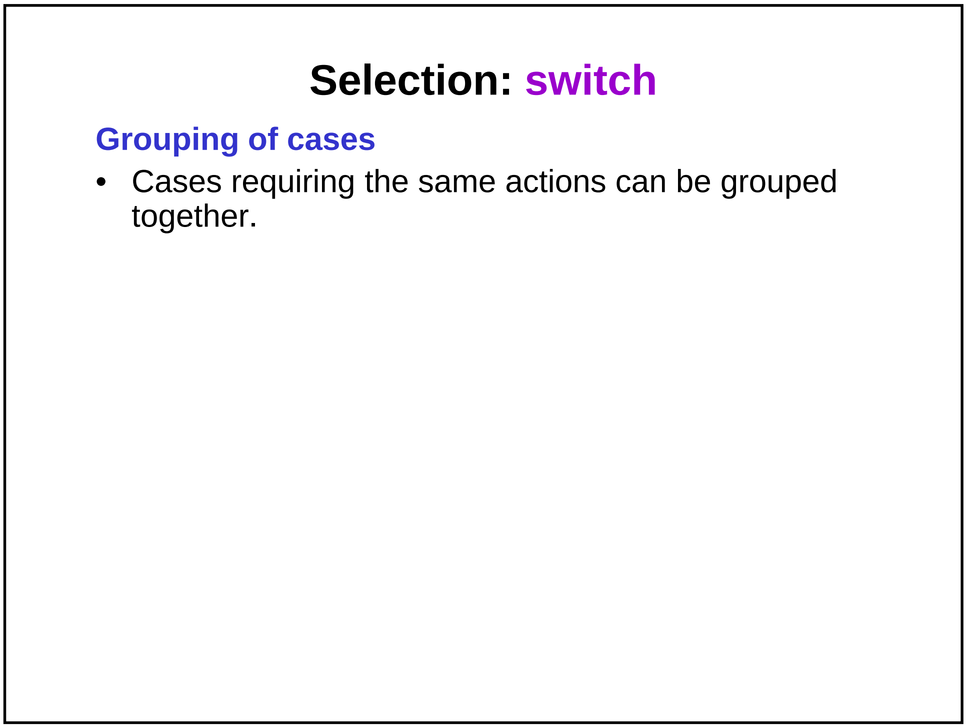
- Input a letter grade, letGrade, and assign the numerical grade equivalent to numGrade.

```
int numGrade;
output.print(“Please enter a letter grade: “);
char letGrade = input.next().charAt(0);
switch (letGrade)
{ case ‘A’ :
       numGrade = 9;
       break;
   case ‘B’ :
        numGrade = 7;
        break;
       // continued on next slide
```

Selection: switch

Example

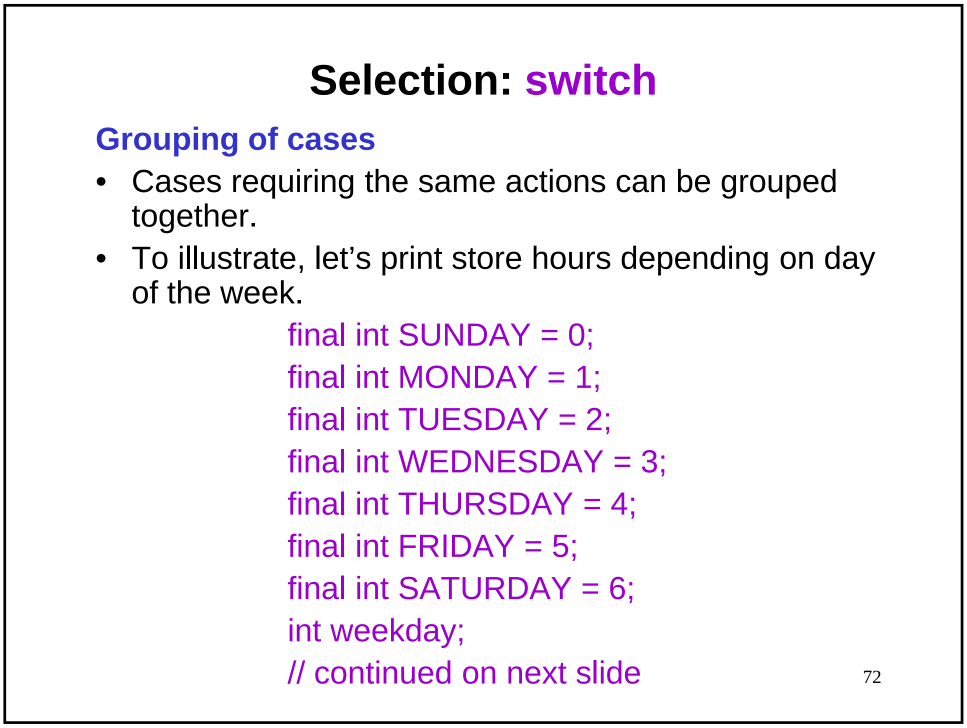
```
// continued from previous slide
case ‘C’ :
    numGrade = 6;
    break;
case ‘D’ :
    numGrade = 5;
    break;
case ‘F’ :
    numGrade = 4;
    break;
 default :
    output.println(“Error: Bad letter grade.”);
    numGrade = 0; // don’t leave this unbound
} // end switch on letGrade
```



Selection: switch

Grouping of cases

- Cases requiring the same actions can be grouped together.



Selection: switch

Grouping of cases

- Cases requiring the same actions can be grouped together.
- To illustrate, let’s print store hours depending on day of the week.

```
final int SUNDAY = 0;
final int MONDAY = 1;
final int TUESDAY = 2;
final int WEDNESDAY = 3;
final int THURSDAY = 4;
final int FRIDAY = 5;
final int SATURDAY = 6;
int weekday;
// continued on next slide
```

Selection: switch

Grouping of cases

```
// continued from previous slide
// assume weekday somehow assigned a value
switch (weekday)
{ case MONDAY : case TUESDAY :
  case WEDNESDAY : case SATURDAY :
     output.println(“Hours are 10AM – 6PM.”);
      break;
  case THURSDAY : case FRIDAY :
      output.println(“Hours are 10AM – 9PM.”);
      break;
  // continued on next slide
```

Selection: switch

Grouping of cases

```
// continued from previous slide
  case SUNDAY :
      output.println(“Closed.”);
      break;
  default:
       output.println(“Error: Bad weekday.”);
} // end switch on weekday
```

Selection: switch

Recapitulation

- The switch statement allows for selection among a set of alternatives.
- We must be able to represent the alternatives as an ordinal type, e.g., int or char.

Outline

- Flow of control
- Selection: if
- Selection: switch
- **Selection: The ternary operator**
- Selection: Recapitulation
- File I/O

Selection: The ternary operator

Selection based on ?

- We have seen that we can select between two possible conditions based on an if construction of the following form.

```
if (condition)
{ x = value1;
} else
{ x = value2;
}
```

Selection: The ternary operator

Selection based on ?

- We have seen that we can select between two possible conditions based on an if construction of the following form.

```
if (condition)
{ x = value1;
} else
{ x = value2;
}
```

- In Java, we can get the same behaviour by using ?

```
x = condition ? value1 : value2;
```

- Essentially, we have a “short hand”, albeit one that is a bit obscure.

Outline

- Flow of control
- Selection: if
- Selection: switch
- Selection: The ternary operator
- **Selection: Recapitulation**
- File I/O

Selection: Recapitulation

A few points to beware

- Do not have your conditionals depend on exact equality of real variables (types float and double).

Selection: Recapitulation

A few points to beware

- Do not have your conditionals depend on exact equality of real variables (types float and double)
- Example: The following code outputs “Not Equal!” for many inputs.

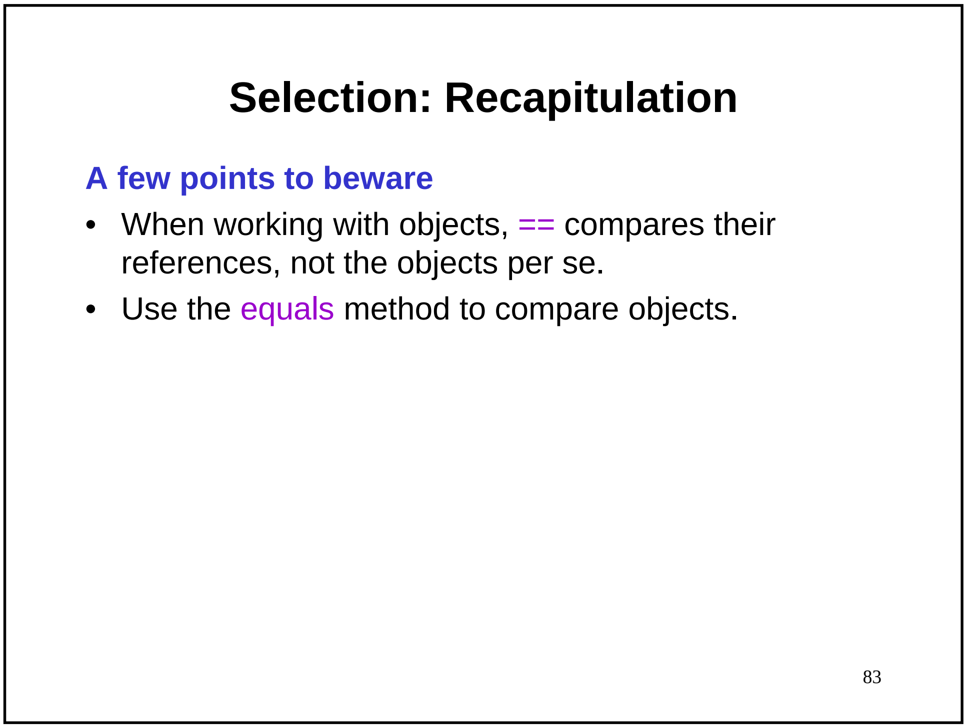
```
output.print(“Enter a real: ”);
double x  = input.nextDouble();
double y = Math.pow(Math.pow(x, 0.5), 2);
if (x == y)
{ output.println(“Equal!”);
} else
{ output.println(“Not equal!”);
}
```

Selection: Recapitulation

A few points to beware

- Do not have your conditionals depend on exact equality of real variables (types float and double)
- Example: A proper way to code the previous example would be as follows.

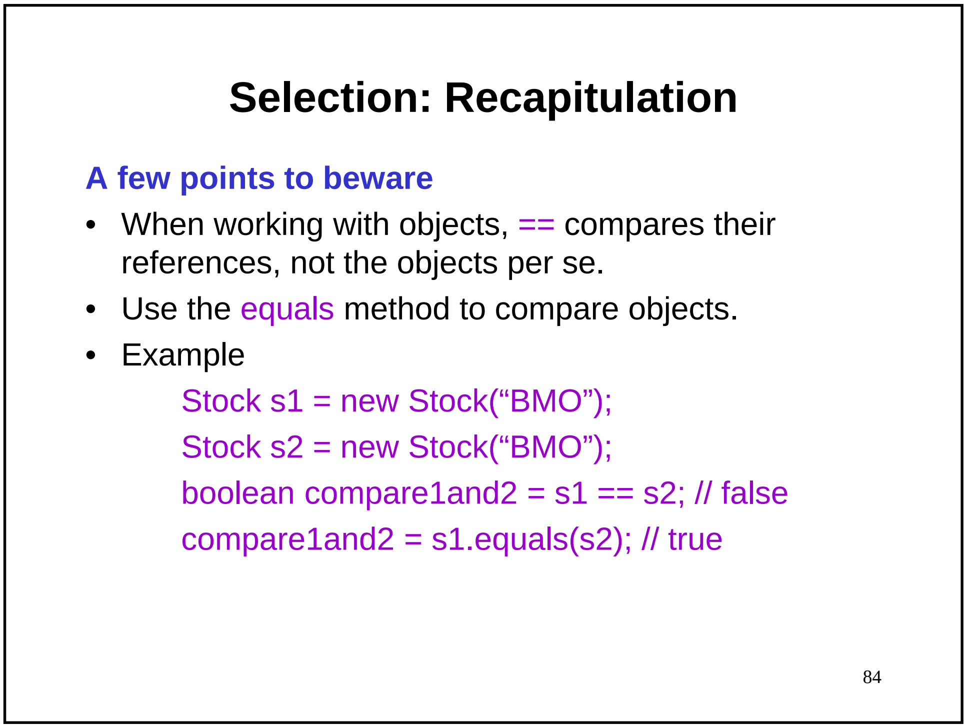
```
output.print(“Enter a real: ”);
double x  = input.nextDouble();
double y = Math.pow(Math.pow(x, 0.5), 2);
if (Math.abs(x – y) < EPSILON) // EPSILON small
{ output.println(“Equal!”);
} else
{ output.println(“Not equal!”);
}
```



Selection: Recapitulation

A few points to beware

- When working with objects, == compares their references, not the objects per se.
- Use the equals method to compare objects.



Selection: Recapitulation

A few points to beware

- When working with objects, == compares their references, not the objects per se.
- Use the equals method to compare objects.
- Example

```
Stock s1 = new Stock("BMO");
Stock s2 = new Stock("BMO");
boolean compare1and2 = s1 == s2; // false
compare1and2 = s1.equals(s2); // true
```

Selection: Recapitulation

A few points to beware

- It is poor style to style to use a conditional to assign a boolean value.

Selection: Recapitulation

A few points to beware

- It is poor style to style to use a conditional to assign a boolean value.
- Example: We could write

```
boolean valid;
if (x>a && y <=b)
{ valid = true;
} else
{ valid = false;
}
```

However, it is much cleaner to write

```
boolean valid = (x>a && y<=b);
```

Selection: Recapitulation

A few points to beware

- Math notation is not acceptable in conditions.

Selection: Recapitulation

A few points to beware

- Math notation is not acceptable in conditions.
- Example: Here is something that will not compile

```
if ((0<x<1) || (x&&y) > 1)  // error
```

presumably, the intent is as follows.

```
if ((x>0 && x<1) || (x>1 && y>1))
```

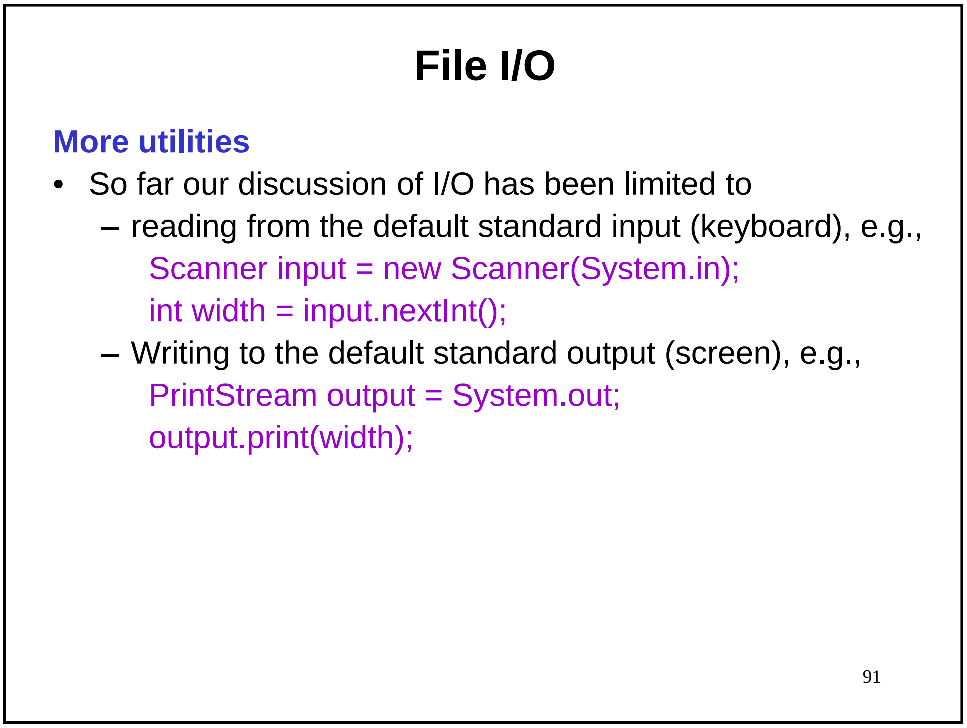
Selection: Recapitulation

Final remarks

- We have seen two ways to select between alternatives.
  1. Based on the truth value of a single boolean: if
  2. Based on scanning against a list of possibilities: switch
- We also have a seen short hand notation for simple selection between two alternatives: ?

Outline

- **Flow of control**
- **Selection: if**
- **Selection: switch**
- **Selection: The ternary operator**
- **Selection: Recapitulation**
- **File I/O**



File I/O

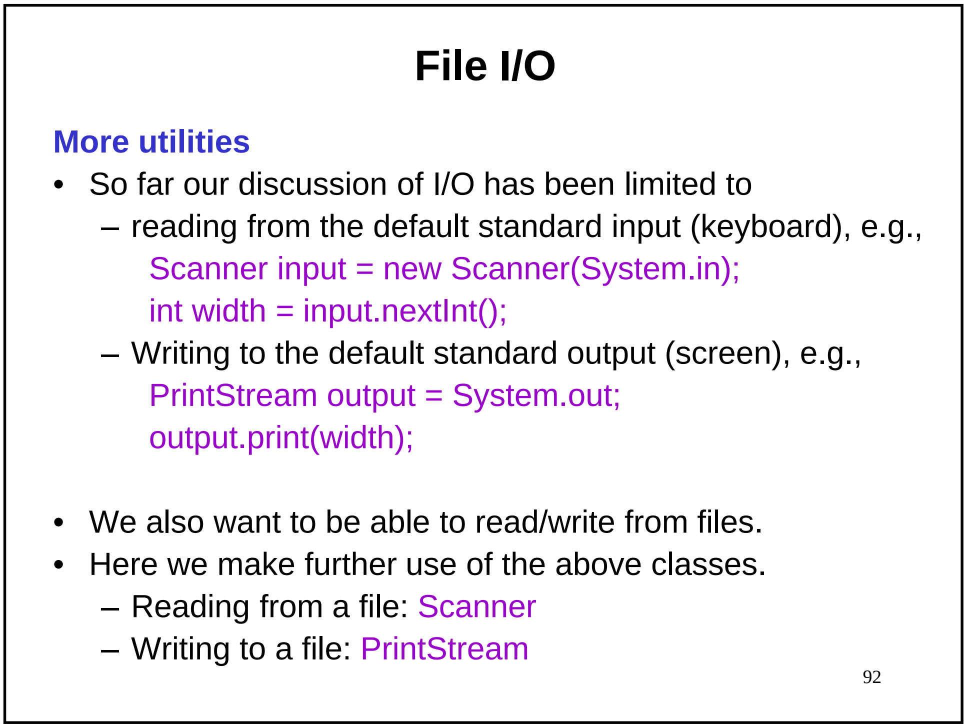
More utilities

- So far our discussion of I/O has been limited to
  - reading from the default standard input (keyboard), e.g.,

```
Scanner input = new Scanner(System.in);
int width = input.nextInt();
```

  - Writing to the default standard output (screen), e.g.,

```
PrintStream output = System.out;
output.print(width);
```



File I/O

More utilities

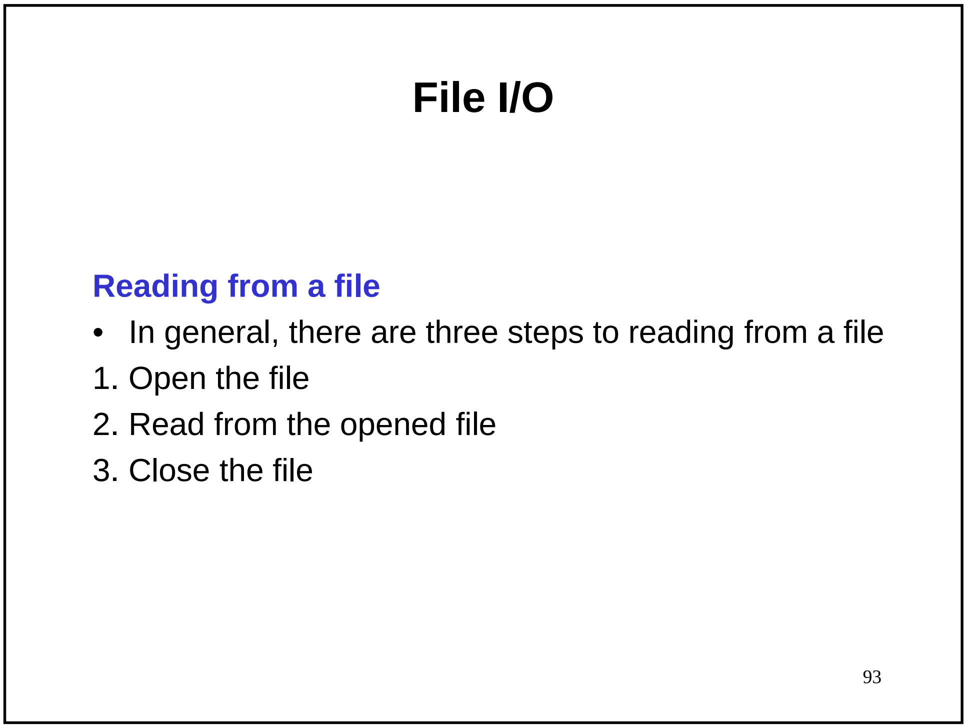
- So far our discussion of I/O has been limited to
  - reading from the default standard input (keyboard), e.g.,

```
Scanner input = new Scanner(System.in);
int width = input.nextInt();
```

  - Writing to the default standard output (screen), e.g.,

```
PrintStream output = System.out;
output.print(width);
```

- We also want to be able to read/write from files.
- Here we make further use of the above classes.
  - Reading from a file: Scanner
  - Writing to a file: PrintStream

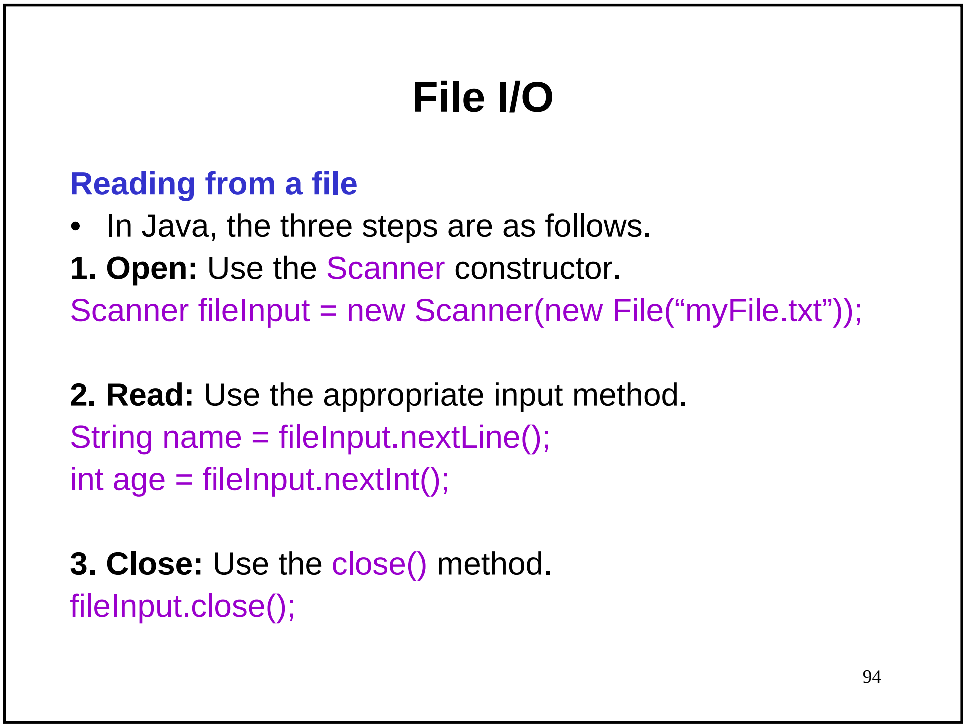


File I/O

Reading from a file

- In general, there are three steps to reading from a file

1. Open the file
2. Read from the opened file
3. Close the file



File I/O

Reading from a file

- In Java, the three steps are as follows.

1. Open: Use the Scanner constructor.

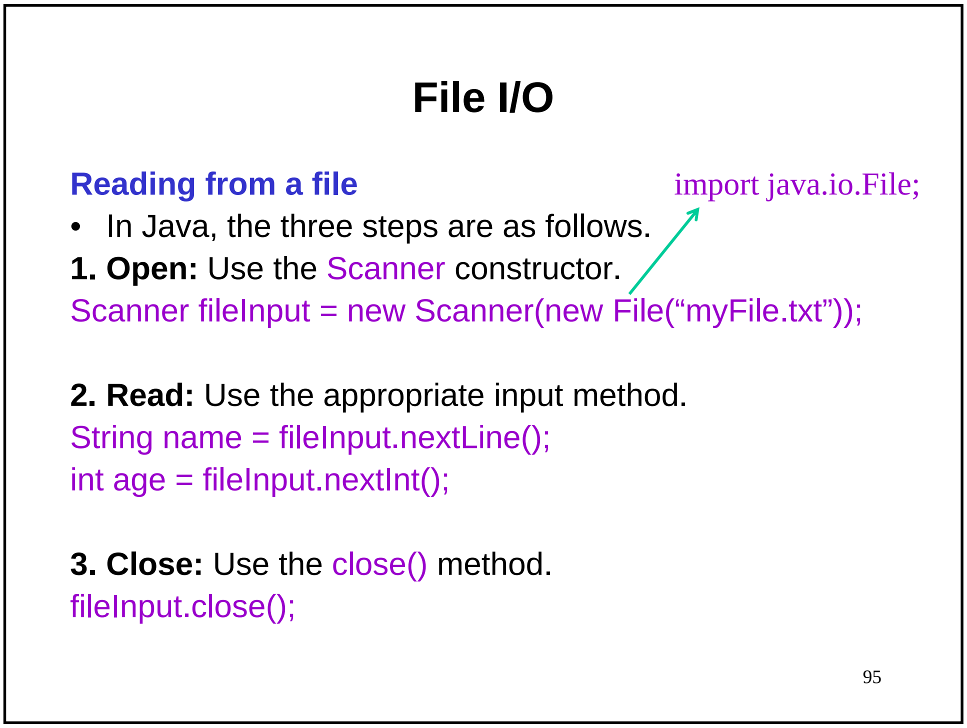
```
Scanner fileInput = new Scanner(new File(“myFile.txt”));
```

2. Read: Use the appropriate input method.

```
String name = fileInput.nextLine();
int age = fileInput.nextInt();
```

3. Close: Use the close() method.

```
fileInput.close();
```



File I/O

Reading from a file

import java.io.File;

- In Java, the three steps are as follows.

1. Open: Use the Scanner constructor.

Scanner fileInput = new Scanner(new File(“myFile.txt”));

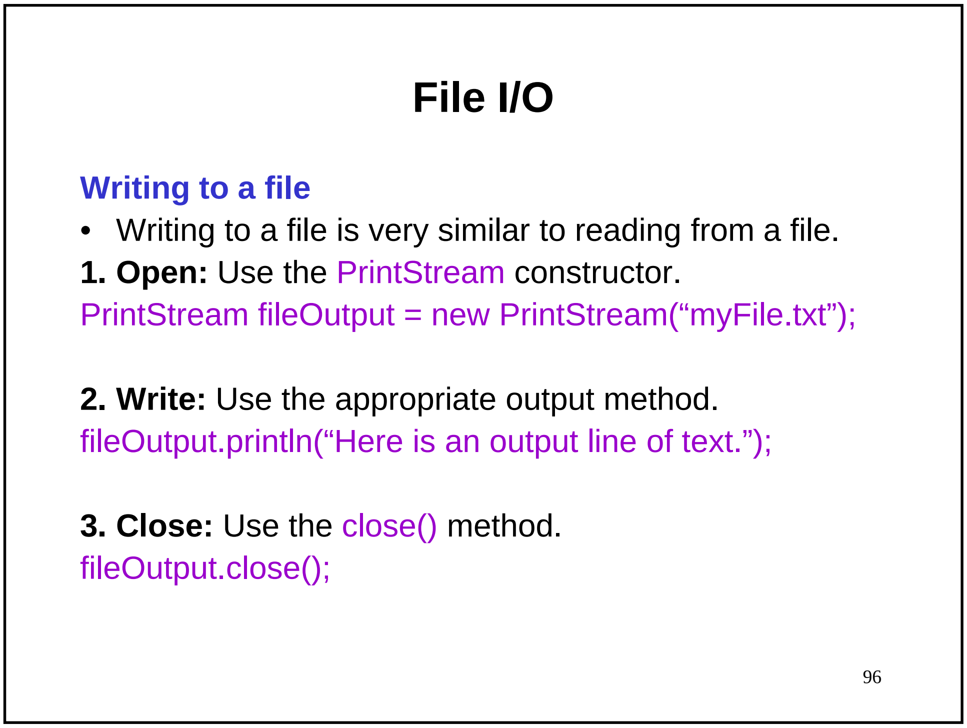
2. Read: Use the appropriate input method.

String name = fileInput.nextLine();

int age = fileInput.nextInt();

3. Close: Use the close() method.

fileInput.close();



File I/O

Writing to a file

- Writing to a file is very similar to reading from a file.

1. Open: Use the PrintStream constructor.

PrintStream fileOutput = new PrintStream(“myFile.txt”);

2. Write: Use the appropriate output method.

fileOutput.println(“Here is an output line of text.”);

3. Close: Use the close() method.

fileOutput.close();

File I/O

A few fine points

- File name can be input from a
  - standard input stream (e.g., a Scanner instance) as a String (we learn about strings in Unit 6).
  - dialog box (see textbook, p. 199).

File I/O

A few fine points

- File name can be input from a
  - standard input stream (e.g., a Scanner instance) as a String (we learn about strings in Unit 6).
  - dialog box (see textbook, p. 199).

- When ever a program deals with files, it is possible that they are not present.
  - An I/O exception should be thrown (we learn about exceptions in Unit 11).
  - To enable the exception we must modify the header of our main method as follows.

```
public static void main(String[] args) throws java.io.IOException
```

File I/O

A few fine points

- File name can be input from a
  - standard input stream (e.g., a Scanner instance) as a String (we learn about strings in Unit 6).
  - dialog box (see textbook, p. 199).
- When ever a program deals with files, it is possible that they are not present.
  - An I/O exception should be thrown (we learn about exceptions in Unit 11).
  - To enable the exception we must modify the header of our main method as follows.

  ```
  public static void main(String[] args) throws java.io.IOException
  ```
- If you need to check that a type P is available for reading, then you can use the hasNextP() method.

File I/O

A few fine points

- File name can be input from a
  - standard input stream (e.g., a Scanner instance) as a String (we learn about strings in Unit 6).
  - dialog box (see textbook, p. 199).
- When ever a program deals with files, it is possible that they are not present.
  - An I/O exception should be thrown (we learn about exceptions in Unit 11).
  - To enable the exception we must modify the header of our main method as follows.

  ```
  public static void main(String[] args) throws java.io.IOException
  ```
- If you need to check that a type P is available for reading, then you can use the hasNextP() method.

- As usual, the API is the best reference.

Summary

- Flow of control
- Selection: if
- Selection: switch
- Selection: The ternary operator
- Selection: Recapitulation
- File I/O