|  |
| --- |
| **Set 8: *if* Statements** |

|  |
| --- |
| **Skill 8.1: Write an *if* statement**  **Skill 8.2: Write an *if-else* statement**  **Skill 8.3: Evaluate numeric expressions with *if-else* statements**  **Skill 8.4: Compare string expressions with *if-else* statements**  **Skill 8.5: Combine multiple *if* statements with *if-else***  **Skill 8.6: Write nested *if-else* statements**  **Skill 8.7: Write *if-else* statements without curly brackets** |

|  |
| --- |
| **Skill 8.1: Predict the outcome of an *if* statement** |

**Skill 8.1 Concepts**

Control structures allow us to control the flow of our programs. The *if* statement is the simplist of these. As implied by the name, an *if* statement will allow something to occur *if* a boolean condition is met.

An *if* statement takes the following form.

|  |
| --- |
| if(boolean expression){         //the code between these curly brackets **will** be executed         //if the boolean expression in parenthesis is true  } |

[**Skill 8.1 Exercise**](https://hpluska.github.io/APCompSciPrinciples/ticketOutTheDoor/set0/Set0TicketOutTheDoorAPCompSciPrinciples.pdf) **1**

|  |
| --- |
| **Skill 8.2: Predict the outcome of an *if-else* statement** |

**Skill 8.2 Concepts**

The *if-else* statement allows you to handle conditions that evaluate to true as well as false. Just as we saw in the example above, the *if* portion of this statement will occur *if* a boolean condition is met, otherwise the *else* portion of the statement will be executed. Consider the example below,

|  |
| --- |
| if(boolean expression){  //the code between these curly brackets **will** be executed  //if the boolean expression in parenthesis is **true**  }else{  //the code between these curly brackets **will** be executed  //if the boolean expression in parenthesis is **false**  } |

[**Skill 8.2 Exercise**](https://hpluska.github.io/APCompSciPrinciples/ticketOutTheDoor/set0/Set0TicketOutTheDoorAPCompSciPrinciples.pdf) **1**

|  |
| --- |
| **Skill 8.3: Evaluate numeric expressions with *if-else* statements** |

**Skill 8.3 Concepts**

n a previous lesson we learned that numerical expressions can be evaluated using either of the following expressions.

==     !=

Consider the following examples,

|  |
| --- |
| int i1 = 2;  int i2 = 3;  if(i1 == i2){  //the code between these curly brackets **will not** be executed  //because the condition (i1 == i2) is **false**  }else{  //the code between these curly brackets **will** be executed  //because the condition (i1 == i2) is **false**  } |

|  |
| --- |
| int i1 = 2;  int i2 = 3;  if(i1 != i2){  //the code between these curly brackets **will** be executed  //because the condition (i1 != i2) is **true**  }else{  //the code between these curly brackets **will not** be executed  //because the condition (i1 != i2) is **true**  } |

Numerical expressions can also be combined using either the OR and AND operators,

||     &&

For examples,

|  |
| --- |
| int i1 = 2;  int i2 = 3;  if((i1 < 10 ) && (i2 == 5)){  //the code between these curly brackets **will not** be executed  //because the condition ((i1 < i2) && (i2 == 5)) is **false**  }else{  //the code between these curly brackets **will** be executed  //because the condition ((i1 < i2) && (i2 == 5)) is **false**  } |

|  |
| --- |
| int i1 = 2;  int i2 = 3;  if((i1 < 10) || (i2 == 5)){  //the code between these curly brackets **will** be executed  //because the condition ((i1 < 10) || (i2 == 5)) is **true**  }else{  //the code between these curly brackets **will not** be executed  //because the condition ((i1 < 10) || (i2 == 5)) is **true**  } |

[**Skill 8.3 Exercise**](https://hpluska.github.io/APCompSciPrinciples/ticketOutTheDoor/set0/Set0TicketOutTheDoorAPCompSciPrinciples.pdf) **1**

|  |
| --- |
| **Skill 8.4: Compare string expressions with *if-else* statements** |

**Skill 8.4 Concepts**

In the previous examples the *==* notation was used to compare values. However, this notation is only valid for comparing primitive type variables. Recall that String variables are **not** primitives - they are *object* variables. We will discuss the difference between primitive and object type variables later, but for now just recognize that String variables (or any object type variable) **cannot** be compared using the *==* operator. String variables can only be compared by using the *equals* method illustrated below,

|  |  |
| --- | --- |
| **Code** | **Output** |
| String c = "cat";  String d = "dog";  boolean b = c.equals(d);//compares d to c and assigns the result to b  System.out.println(b);  b = !c.equals(d);//reverses the comparison with the not operator System.out.println(b); | False  True |

How the equals method can be combined with *if-else* statements is illustrated below,

|  |
| --- |
| String s1 = "ant";  String s2 = "ark";  String s3 = "ark";  if(( s1.equals(s2)) && ( s2.equals(s3))){  //the code between these curly brackets **will not** be executed  //because the condition ((s1.equals(s2)) && (s2.equals(s3))) evaluates to **false**  }else{  //the code between these curly brackets **will** be executed  //because the condition ((s1.equals(s2)) && (s2.equals(s3))) evaluates to **false**  } |

|  |
| --- |
| String s1 = "ant";  String s2 = "ark";  String s3 = "ark";  if(( s1.equals(s2)) || ( s2.equals(s3))){  //the code between these curly brackets **will** be executed  //because the condition ((s1.equals(s2)) || (s2.equals(s3))) evaluates to **true**  }else{  //the code between these curly brackets **will not** be executed  //because the condition ((s1.equals(s2)) || (s2.equals(s3))) evaluates to **true**  } |

[**Skill 8.4 Exercise**](https://hpluska.github.io/APCompSciPrinciples/ticketOutTheDoor/set0/Set0TicketOutTheDoorAPCompSciPrinciples.pdf) **1**

|  |
| --- |
| **Skill 8.5: Combine multiple *if* statements with *if-else*** |

**Skill 8.5 Concepts**

Multiple *if* statements can be combined with *if-else*. This is illustrated below,

|  |
| --- |
| Scanner s = new Scanner(System.in);  System.out.println("What is your grade?");  int theGrade = s.nextInt();  if(theGrade>=90){      System.out.println("You got an A!");  }else if(theGrade>=80){      System.out.println("You got a B!");  }else if(theGrade>=70){      System.out.println("You got a C!");  }else if(theGrade>=60){      System.out.println("You got a D!");  }else{      System.out.println("You failed!");  } |

In the example above, we wrote the code with the most restrictive option first. This eliminated the need to check for two boundary conditions. This is a good rule of thumb for both simplifying the logic reducing unexpected outcomes.

[**Skill 8.5 Exercise**](https://hpluska.github.io/APCompSciPrinciples/ticketOutTheDoor/set0/Set0TicketOutTheDoorAPCompSciPrinciples.pdf) **1**

|  |
| --- |
| **Skill 8.6: Write nested *if-else* statements** |

**Skill 8.6 Concepts**

A nested *if* or *if-else* statement refers to an *if* or *if-else* statement inside of an *if-else* statement. Consider the following example,

|  |
| --- |
| Scanner s = new Scanner(System.in);  System.out.println("What is your grade?");  int theGrade = s.nextInt();  if(theGrade>=90){      if(theGrade<94){          System.out.println("you got an A-");      }else{          System.out.println("you got an A");      }  }else if(theGrade>=80){      if(theGrade>86){          System.out.println("you got a B+");      }else if(theGrade<84){          System.out.println("you got a B-");      }else{          System.out.println("you got a B");      }  }else{      System.out.println("you got another grade");  } |

[**Skill 8.6 Exercise**](https://hpluska.github.io/APCompSciPrinciples/ticketOutTheDoor/set0/Set0TicketOutTheDoorAPCompSciPrinciples.pdf) **1**

|  |
| --- |
| **Skill 8.7: Write *if-else* statements without curly brackets** |

**Skill 8.7 Concepts**

Curly brackets are not needed if only one line of code is in the *if* or *else* parts. Likewise, the absence of brackets implies only one line of code in the *if-else* parts.

This is illustrated below.

|  |
| --- |
| **Code** |
| int groovyDude=37;  if(groovyDude == 37)      groovyDude++;//only this line is executed      System.out.println(groovyDude);//this line is always printed |
| **Output** |
| 38 |

|  |
| --- |
| **Code** |
| int groovyDude=100;  if(groovyDude == 37)  groovyDude++;//only this line is not executed      System.out.println(groovyDude);//this line is always printed |
| **Output** |
| 100 |

|  |
| --- |
| **Code** |
| int groovyDude=100;  if(groovyDude == 37)  groovyDude++;//only this line is not executed      System.out.println(groovyDude);//this line is always printed |
| **Output** |
| 100 |

|  |
| --- |
| **Code** |
| int groovyDude=37;  if(groovyDude == 37)  groovyDude++;//this line will execute  else      groovyDude--;//this line will not execute      System.out.println(groovyDude);//this line will execute |
| **Output** |
| 38 |

The above examples illustrate the importance of indention when writing if statements. Always indent the code associated with the logical statement for readability. The examples above are re-written below to illustrate this point.

|  |
| --- |
| **Code** |
| int groovyDude=37;  if(groovyDude == 37)      groovyDude++;//only this line is executed  System.out.println(groovyDude);//this line is always printed |
| **Output** |
| 38 |

|  |
| --- |
| **Code** |
| int groovyDude=100;  if(groovyDude == 37)  groovyDude++;//only this line is not executed  System.out.println(groovyDude);//this line is always printed |
| **Output** |
| 100 |

|  |
| --- |
| **Code** |
| int groovyDude=100;  if(groovyDude == 37)  groovyDude++;//only this line is not executed  System.out.println(groovyDude);//this line is always printed |
| **Output** |
| 100 |

|  |
| --- |
| **Code** |
| int groovyDude=37;  if(groovyDude == 37)  groovyDude++;//this line will execute  else      groovyDude--;//this line will not execute  System.out.println(groovyDude);//this line will execute |
| **Output** |
| 38 |

[**Skill 8.7 Exercise**](https://hpluska.github.io/APCompSciPrinciples/ticketOutTheDoor/set0/Set0TicketOutTheDoorAPCompSciPrinciples.pdf) **1**