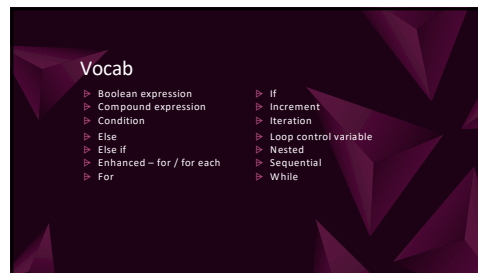
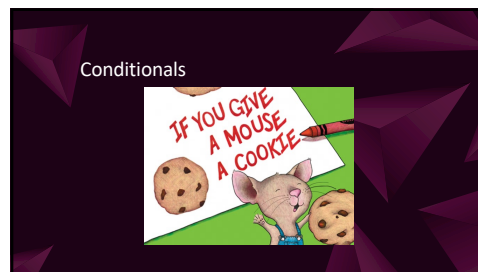




1



2



3

If

- The default if block is only evaluated if and only if the associated condition evaluates as true
- When failed the code drops to below the closing squiggle
- When comparing Boolean values do NOT use `== true` or `== false` just use the variable or the negated variable

4

Else if

- You failed the previous if and want to test an ADDITIONAL condition
- Make sure the condition is possible after the previous condition

5

Else

- This block only executes if and only if the previous if condition was failed
- Can never enter a else if a previous if was true
- The LAST option in a sequence of if/else if/...

6

Sequential if blocks

- If there are no else blocks, each individual if is ALWAYS evaluated
- Use else blocks to skip evaluations

7

Nested Conditionals

- Evaluation only continues when the conditions continue to be true
- Can also be designed as a compound condition (using `||` / `&&`)
- Stylistic choice
- Make code easier to read for humans, not computers!
- No more than two compound Boolean expressions is my style

8

Conditional Examples

```
if (hasHose)
    System.out.println("The backhoeer has a hose!");
else
    System.out.println("The backhoeer has no hose!");
```

```
if (hasHose)
{
    System.out.println("yes hose!");
}
else
{
    System.out.println("no hose!");
}
```

```
if (backhoeer.getArmCount() > 100)
{
    System.out.println("Lots of arms!");
}
else if (backhoeer.getArmCount() > 50)
{
    System.out.println("Not so many arms but a lot!");
}
else if (backhoeer.getArmCount() > 0)
{
    System.out.println("There are arms!");
}
else
{
    System.out.println("No arms!");
}
```


9

Nested Conditional Example

```
boolean isMonster = isMonster().getIsMonster();
if (isMonster)
{
    (isMonster().getIsScary())
    {
        System.out.println("The monster has a nose but isn't scary");
    }
}
```

10

Loops



11

While loop

- ▷ Undefined execution time
- ▷ Execute until failure
- ▷ Loop control variable has scope BEYOND the loop
- ▷ Update statement SHOULD be the last statement of the loop body

12

While Example

```

boolean initialized = false;
int count = 0;
while (!initialized)
{
    System.out.println("Keep on going!!!");
    count++;
    if (count > 1000)
    {
        initialized = true;
    }
}

while (initialized && count < 50)
{
    System.out.println("The amount of " + MarshallInfo.getName());
    MarshallInfo.setAmountCount(MarshallInfo.getAmountCount() + 1);
}

```

13

Standard for loop

- ▶ Counting loop
- ▶ Three part header
 - ▷ Initialization
 - ▷ Condition
 - ▷ Incrementation
- ▶ Perfect for manipulating structures/contents
- ▶ Index based access
- ▶ Direction and increment are modifiable
- ▶ Loop control variable only has scope in the loop

14

For Examples

```

String name = MarshallInfo.getName();
for (int count = 0; count < MarshallInfo.getCount(); count++)
{
    String letter = name.substring(count, count + 1);
    System.out.println(letter);
}

for (int count = 0; count < MarshallInfo.getCount(); count++)
{
    MarshallInfoMaster MarshallInfoMaster = MarshallInfo.get(count);
    String name = MarshallInfoMaster.getName();
    System.out.println("The master at this spot has a name of: " + name);
}

for (int count = MarshallInfo.getCount() - 1; count >= 0; count--)
{
    MarshallInfoMaster MarshallInfoMaster = MarshallInfo.get(count);
    String name = MarshallInfoMaster.getName();
    System.out.println("The master at " + count + " has " + MarshallInfo.getAmount() + " area");
}

```

15

Enhanced For AKA For-Each

- ▶ Structure based loop
 - ▷ ArrayList, array, or any other CollectionType
 - ▷ A String is **NOT** a structure
- ▶ Two part header
 - ▷ Type variableName : structure
- ▶ Does the same operation to **EVERY** item in the structure
- ▶ No access to index
- ▶ Cannot replace, remove, or add the contents of the structure
 - ▷ Throws a ConcurrentModificationException
- ▶ Can use the dot operator to modify internal state of **objects** in the structure
 - ▷ AKA use the setter methods!
- ▶ Order is based on the default iterator of the structure
 - ▷ Direction cannot be controlled

16

For-Each example

```
for (MarshmallowMonster monster : firstList)
{
    System.out.println("The arm count is: " + monster.getArmCount());
}
```

17

Nested loops

- ▶ The inner loop does a complete set of executions for every individual step of the outer loop
- ▶ Multiply the inner loop execution count times the outer loop execution count to see how many times the inner loop executes
- ▶ Standard approach for many algorithms
 - ▷ Sorting lists and arrays
 - ▷ Manipulating data
 - ▷ Multi-dimensional arrays

18

Nested Loop example

```
for (int outer = 0; outer < 10; outer++)  
{  
    System.out.println("This is the outer loop");  
    for (int inner = 0; inner < 20; inner++)  
    {  
        System.out.println("This is the inner loop: " + outer + ", and index (outer): " + inner);  
    }  
    System.out.println("Outer loop completed @ " + outer);  
}
```

19
