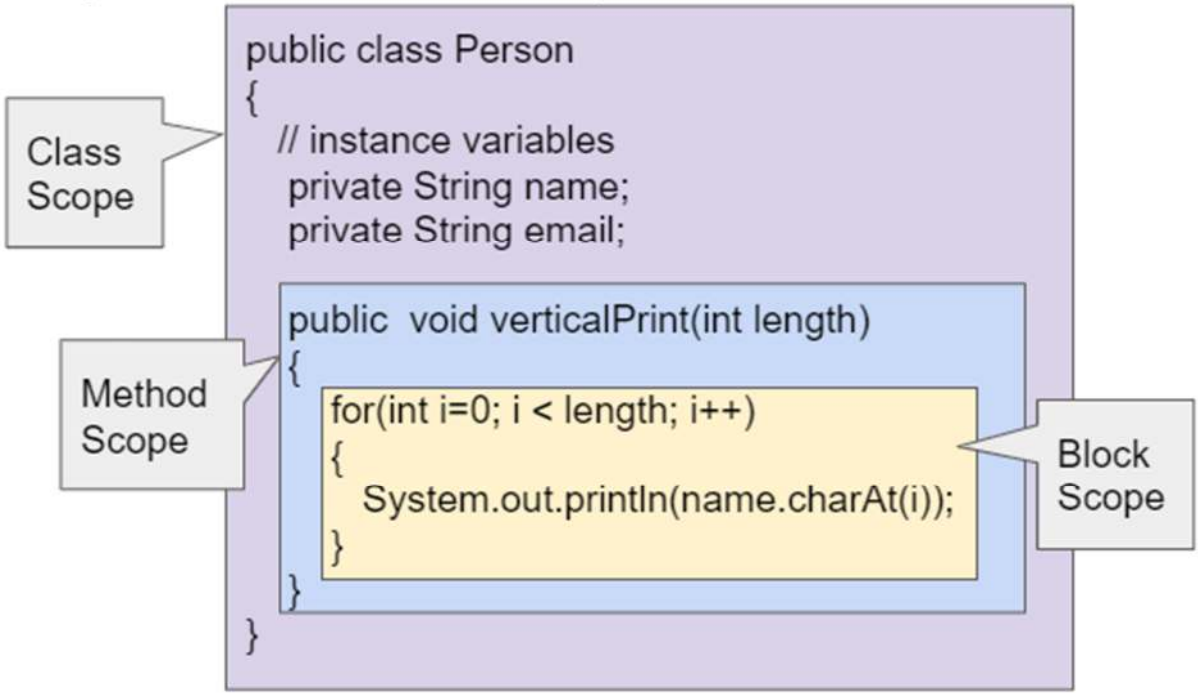
Scope and Access

The **scope** of a variable is defined as where a variable is accessible or can be used. The scope is determined by where you declare the variable when you write your programs. When you declare a variable, look for the closest enclosing curly brackets `{ }` – this is its scope.

Java has 3 levels of scope that correspond to different types of variables:

- **Class Level Scope** for **instance variables** inside a class.
- **Method Level Scope** for **local variables** (including **parameter variables**) inside a method.
- **Block Level Scope** for **loop variables** and other local variables defined inside of blocks of code with `{}`.

The image below shows these 3 levels of scope.



Class, Method, and Block Level Scope

```
public class Name {

private String first;
public String last;
```

```
public Name(String theFirst, String theLast) {
String firstName = theFirst;
first = firstName;
last = theLast;
}
}
```

Click on all the variable declarations that are at **Method** Level Scope.Remember that the parameter variables and the local variables declared inside a method have Method Level Scope.

```
private String first;
public String last;

public Name(String theFirst, String theLast) {
String firstName = theFirst;
first = firstName;
last = theLast;
}
}
```

#Local Variables

Local variables are variables that are declared inside a method, usually at the top of the method. These variables can only be used within the method and do not exist outside of the method. Parameter variables are also considered local variables that only exist for that method. It's good practice to keep any variables that are used by just one method as local variables in that method.

Instance variables at class scope are shared by all the methods in the class and can be marked as public or private with respect to their access outside of the class. They have Class scope regardless of whether they are public or private.

Another way to look at scope is that a variable's scope is where it lives and exists. You cannot use the variable in code outside of its scope. The variable does not exist outside of its scope, but can be used inside of methods that share its scope.

#Coding Exercise

Try the following code to see that you cannot access the variables outside of their scope levels in the `toString()` method. Reach out to your team on Slack and explain to a teammate why you can't access these variables. Talk it through and answer each others' questions, then come back here. Try to fix the errors in the code by either using variables that are in scope or moving the variable declarations so that the variables have the required scope to function.

Run

Person.java

```
public class Person {

private String name;
private String email;

public Person(String initName, String initEmail) {
   name = initName;
   email = initEmail;
}

public String toString() {
  for (int i=0; i < 5; i++) {
     int id = i;
  }
  // Problem 1: Variable 'i' and 'id' are defined within the for loop's sco
pe.
  // You can't access them outside the loop.
  System.out.println("i at the end of the loop is " + i);
  System.out.println("The last id is " + id);  // Compilation error here

  // Problem 2: 'initName' and 'initEmail' are constructor parameters,
  // they are not accessible within the toString() method's scope.
  return  initName + ": " + initEmail;  // Compilation error here
}

public static void main(String[] args) {
   Person p1 = new Person("Sana", "sana@gmail.com");
   System.out.println(p1);
}
}
```

If there is a local variable with the same name as an instance variable, the variable name will refer to the local variable instead of the instance variable, as seen below.

We'll see in the next lesson, that we can distinguish between the local variable and the instance variable using the keyword `this` to refer to this object's instance variables.

Run

Person.java

```
public class Person {
   private String name;
   private String email;

   public Person(String initName, String initEmail) {
      name = initName;
      email = initEmail;
   }

   public String toString() {
     String name = "unknown";
     // The local variable name here will be used,
     //  not the instance variable name.
     return  name + ": " + email;
   }

   // main method for testing
   public static void main(String[] args) {
      // call the constructor to create a new person
      Person p1 = new Person("Sana", "sana@gmail.com");
      System.out.println(p1);
   }
}
```

CONSOLESHELL

#Programming Challenge : Debugging

Debug the following program that has scope violations. Then, add comments that label the variable declarations as class, method, or block scope.

Run

TesterClass.java

```
public class TesterClass {
   public static void main(String[] args) {
      Fraction f1 = new Fraction();
```

```
      Fraction f2 = new Fraction(1,2);
      System.out.println(f1);
      System.out.println(f2.numerator / f2.denominator);
   }
 }
/** Class Fraction */
class Fraction {
   //  instance variables
   private int numerator;
   private int denominator;
   // constructor: set instance variables to default values
```

CONSOLESHELL

#Practice

Consider the following class definitions. Which of the following best explains why the class will not compile?

```
public class Party {
private int boxesOfFood;
private int numOfPeople;

public Party(int people, int foodBoxes) {
numOfPeople = people;
boxesOfFood = foodBoxes;
}

public void orderMoreFood(int additionalFoodBoxes) {
int updatedAmountOfFood = boxesOfFood + additionalFoodBoxes;
boxesOfFood = updatedAmountOfFood;
}

public void eatFoodBoxes(int eatenBoxes) {
boxesOfFood = updatedAmountOfFood - eatenBoxes;
}
}
```

Consider the following class definition.

```
public class Movie
{
private int currentPrice;
private int movieRating;
```

```
public Movie(int p, int r)
{
currentPrice = p;
movieRating = r;
}

public int getCurrentPrice()
{
int currentPrice = 16;
return currentPrice;
}

public void printPrice()
{
System.out.println(getCurrentPrice());
}
}
```

Which of the following reasons explains why the printPrice method is not functioning as intended and only ever prints out a value of `16`?

#Summary

- **Scope** is defined as where a variable is accessible or can be used.
- Local variables can be declared in the body of constructors and methods. These variables may only be used within the constructor or method and cannot be declared to be public or private.
- When there is a local variable with the same name as an instance variable, the variable name will refer to the local variable instead of the instance variable.
- Formal parameters and variables declared in a method or constructor can only be used within that method or constructor.