[the academy_of_code]

Grade ?

Unit Problem Solving

Lesson 1 - Method Overloading

Learning Outcomes:

- Understand what is and how to use method overloading

REMEMBER:

Put up your hand. We love to help!

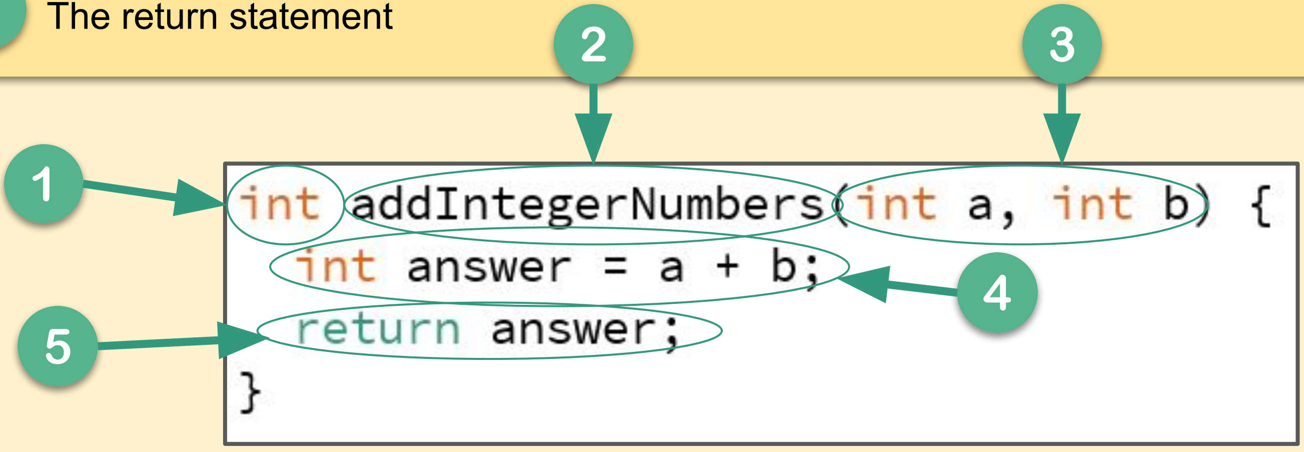
Welcome Back!

In these lessons we are going to learn how to write functions that are more flexible and are able to be used in different ways.

What we need to know

- We need to first remember what a function looks like and what it requires. You should remember from previous lessons the five parts listed below and what they do.

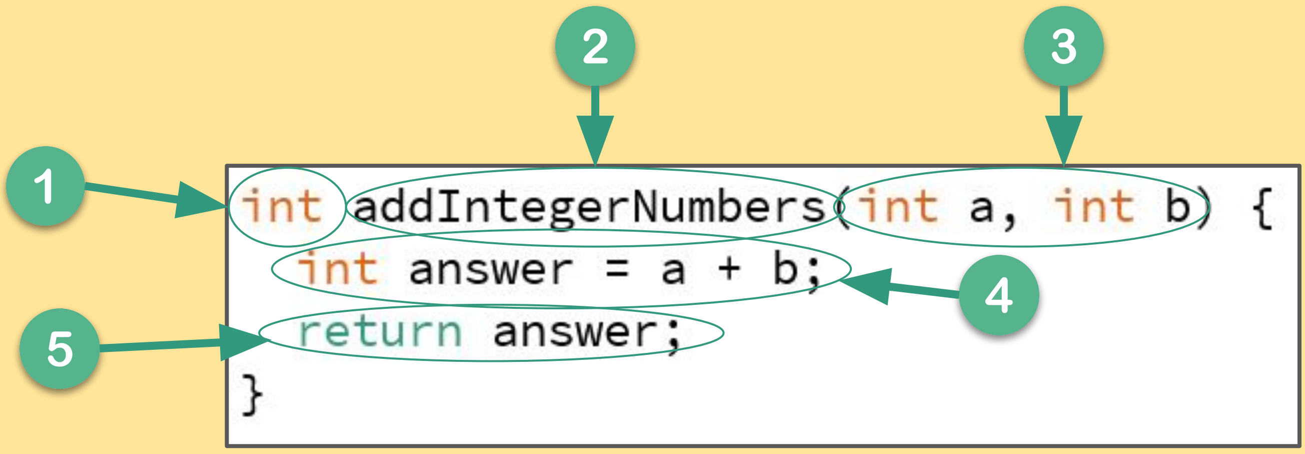
1. Return type of the function
2. Name of the function
3. The parameter list
4. The main code of the function
5. The return statement



```
int addIntegerNumbers(int a, int b) {
  int answer = a + b;
  return answer;
}
```

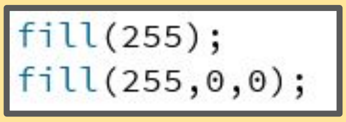
Method Overloading

The part we are most interested in today is number 3, the parameter list. Overloading a function means that we can put a different number of parameters in it and have it run different code.



```
int addIntegerNumbers(int a, int b) {
  int answer = a + b;
  return answer;
}
```

The first example you would have encountered of this is fill(g) vs fill(r, g, b). The first function will give a greyscale colour while the second gives a colour in rgb.



```
fill(255);
fill(255,0,0);
```

When creating an overloaded function we need to be careful to follow certain rules, to make sure that the computer can determine which code we want to run.

Overloading a function is done by either;

- Changing the number of arguments/parameters
- Or changing the datatype of the arguments/parameters
- Or changing the order of the parameters

Important: You can't overload a function by changing the name of the parameters **or** by changing the return type.

Method Overloading Example

The example code below shows a function addNumbers. The original function takes two parameters and adds them together. The overloaded function **changes the number of parameters** and the code is slightly different because of this.

```
int addNumbers(int a, int b) {
  return a+b;
}

int addNumbers(int a, int b, int c) {
  return a+b+c;
}
```

In the second code example we have changed the datatypes of the arguments. We have also changed the name of the arguments, but as noted previously just doing this wouldn’t be enough.

```
int addNumbers(String num1, String num2) {
  return Integer.parseInt(num1) + Integer.parseInt(num2);
}
```

The final example shows that the code can sometimes be different, depending on the parameters.

```
int addNumbers(double a, double b) {
  println("This calculation may result in loss of percision");
  return (int)(a+b);
}
```

Tasks

1. Create a function that will multiply two numbers. Overload the function to be able to multiply three numbers.

2. Create a function that accepts a variable and prints out the variable type.

3. Create a function that if given a number will say that you are that many years old and if given a string will say your name is the entered string.

4. Create a simple class that draw an ellipse. Create three constructors for it. The first doesn't take any parameters and places a circle in a random position. The second takes two numbers and places a circle in that position. The third takes three numbers, the first two are the position and the third is the size of the circle.

5. Create a method for your class that will determine the colour of your circle. It should be able to take a single number for black and white, three numbers for rgb or four numbers for rgba.

6. Create a new class that just stores three values, an x position, a y position and a size. Create an object of this class type and use it in the constructor of the previous class. **Note:** You will need to write a new constructor for the ellipse class.