Lambda Functions in Java

How we declare a function in a class:

```
public void perform() {
 System.out.println("hello");
}
```

Convert that to a Lambda Function

- public is not need outside of a class, so that is dropped.
- name of the function is replaced by the variable it is assigned to, so that is dropped.
- the compiler figures out the return type from the code, so that is dropped.

Which results in:

```
aBlockOfCode = () -> {
  System.out.println("hello");
};
```

Where ( ) contains the arguments for the function.

If your function only contains 1 line you can remove the curly brackets:

```
aBlockOfCode = () -> System.out.println("hello");
```

If your function only contains 1 line you can remove the curly brackets:

Here is a lambda function using parameters.

```
doubleNumberFunction = (int a) -> return a*2;
```

For functions with only 1 line, we can drop the return statement and just have:

```
doubleNumberFunction = (int a) -> a*2;
```

More examples

```
addFunction = (int a, int b) -> a+b;

safeDivideFunction = (in a, int b) -> {
  if(b==0) return 0;
  return a/b;
};
```

If we want to assign a lambda function to a variable like we have above, we will need to create an interface that describes the lambda function. **This interface can have only 1 method.**

```
{
  ...
  MyAdd addFunction = (int a, int b) -> a+b;
  ...
}

interface MyAdd{
  int add(int a, int b);
}
```

References

Reference	URL
Java 8 Lambda Basics 6 - Introducing Lambda Expressions	https://www.youtube.com/watch?v=nUIAvs4OEkM