```
System.out.println(v)
```

Takes a value v and prints it. v can be any value – number, String, etc.

Example.java

```
class Example {
  public static void main(String[] args) {
    int x = 10;
    int y = x + 40;
    System.out.println(x);
    System.out.println(y);
  }
}
```

```
javac and java
```

To run Java programs, we make use of the terminal (the same tool you use to start IDLE3 in the labs) and run the javac and java commands. We will focus on **printed output and assertions** to test our Java programs.

At terminal

```
❯ javac Example.java
❯ java Example
```

```
class ClassName {
  public static void main(String[] args) {
    // Our code starts running here
  }
}
```

Java code (we will write) is always in a file.

There should always be a `class` that has the same name as the file – `ClassName` is defined in the file `ClassName.java`.

Within a few weeks we'll understand what `String[]`, `public`, `static`, and `void` mean, but Java requires that we use all of them with a `main` method to get a program running so we have to shove them in here with little explanation to get started. Sorry.

```
int x = 10;
```

Variable definitions in Java come with an extra piece – an **explicit type annotation** that tells Java what datatype will be stored in the variable. Java enforces that only values of that type can be stored in the variable (in Joe's opinion this is a good thing!)

Example2.java

```
class Example2 {
  static int square(int n) {
    return n * n;
  }
  public static void main(String[] args) {
    System.out.println("Should be 100: ");
    System.out.println(square(10));
    assert square(10) == 100;
    assert square(5) == 25;
  }
}
```

At terminal

```
❯ javac Example2.java
❯ java -ea Example2   # ea means "enable assertions"
```

```
// Python function
def square(n):
  return n * n
```

```
// Java static method
static int square(int n) {
  return n * n;
}
```

The closest thing to a **function definition** from Python is a **static method definition** in Java.

The key differences are the keyword static and that each argument also has an **explicit type annotation**.

StringExample.java

```
class StringExample {
  static String shout(String s) {
    return s.toUpperCase() + "!";
  }
  static String eTo3(String s) {
    return s.replace("e", "3");
  }
  public static void main(String[] args) {
    System.out.println(shout("a"));
    System.out.println(eTo3("hello"));

    assert shout("hi").equals("HI!");
    // Use .equals() with strings

    assert shout("hi") == "HI!";
    // This line should not use ==!
  }
}
```

At terminal

```
❯ javac StringExample.java
❯ java -ea StringExample
```

`String.toUpperCase()`
Returns the string in uppercase (similar to `upper()` in Python)

`String.equals(String s)`
Returns true if the string s has the same characters as this string.

`==` vs. `.equals()`

== is untrustworthy on Strings in Java, and we'll get into the details of why. But we typically should **not** compare strings for equality with == in Java, but instead use the .equals() method.

Example3.java

```java
class Example3 {
  public static void main(String[] args) {
    int x = 10;
    int y = x + 40;
    System.out.println("We expect y to be 50: " + y);
  }
}
```

At terminal

```
❯ javac Example3.java
❯ java Example3
```

Example4.java

```java
class Example4 {
  static int averageOfTwoInt(int n1, int n2) {
    return (n1 + n2) / 2;
  }
  static double averageOfTwoFloat(double n1, double n2) {
    return (n1 + n2) / 2;
  }
  public static void main(String[] args) {
    System.out.println(averageOfTwoInt(4, 5));
    System.out.println(averageOfTwoFloat(4.0, 5.0));
  }
}
```

At terminal

```
❯ javac Example4.java
❯ java Example4
```

IsLonger.java

```java
class IsLongerThan {


  static _____________ isLongerThan(________________________________________) {






  }
  public static void main(String[] args) {
    System.out.println(isLongerThan("abc", 4));
    System.out.println(isLongerThan("abc", 2));
    System.out.println(isLongerThan("password", 7));
  }
}
```

`String.length()`

Returns the number of characters in this string

PA1.java

```java
class PA1 {


  static _____________ convertAndCompare(____________________________________) {






  }
  public static void main(String[] args) {




  }
}
```

Challenge: Reproduce your convertAndCompare program from PA1 into a Java program!

Use println to show the results, and define convertAndCompare as a static method