Non-static features of classes

Pass-by-value or pass-by-reference

There are two ways variables/fields are stored in Java

* Value – the variable name actually represents the value of the variable
* Reference – the variable contains the address in memory where the variable lives

In Java, all object types are stored by reference

BankAccount b;

b = new BankAccount();

BankAccount c = b;

What the above code will do is create two different pointers to the SAME object

Not all variables in Java are of object type – there are several that are of “primitive” type, and these are stored by value

int, double, char, …

… primitive data types are normally keywords

When we pass variables as arguments to a method:

* pass an object: pass by reference
* pass a primitive: pass by value

How do we create a true copy of an object?

* clone()
* Copy constructor

If you are inside e.g. BankAccount, and you have access to any other object of type BankAccount, you can access all the private fields directly.

Equals and toString

All objects have default implementations of Equals and toString

* equals() checks to see if two objects are equal
* toString() transforms the object into a string for printing to the screen (super useful for debugging)

equals:

BankAccount b = new BankAccount(5);

BankAccount c = new BankAccount(b);

System.out.println(b==c);