Inherited methods of classes

* toString()
* equals()
* hashCode()
* getClass()
* a bunch of others which are not used at this stage
* All of these methods are inherited from the default type Object – you can use any of them in your class, or you can override them with your own implementation

getClass()

* tells you what class the object belongs to
* better than instanceof because instanceof gets confused under inheritance
	+ instanceof tells you whether you can cast
	+ it doesn’t tell you what class the object belongs to

in an equals() method, first check:

* argument object o is null
* o.getClass() != this.getClass()

If either is true, then equals() should return false

hashCode()

* A hash is a special number that lets you check whether two objects are unique

Default declaration:

public int hashCode()

* Useful in declaring hash sets
* Hashes are also useful in quickly figuring out if two data strings are equal
* Are hashes ALWAYS unique?
* There is the possibility that two hashes will “collide”
* BankAccount:
	+ int accountNumber
	+ double balance
	+ Number of possible combinations:
		- 2^32 ints
		- 2^64 doubles
		- 2^96 possibilities
	+ but hashCode returns int
		- 2^32 ints
	+ collisions are okay, but you should work to make sure typical data values have unique hashes

Default hashcode: returns a unique identifier for each object which is set when the object is created – has nothing to do with the internal data

* Definitely, hashcodes should be equal if equals() returns true
* Most of the time, hashchodes should be unequal if equals() returns false (can’t always be true because of collisions) – should be true for typically occurring data

Here’s a decent way to generate a hashcode:

* create a string with all the unique data that you need (perhaps this can be obtained from toString()) – call this string s
* return s.hashCode() – calls the string version of hashCode which generates a sophisticated has