The Object Class

- Every java class has Object as its superclass and thus inherits the Object methods.
- Object is a non-abstract class
- Many Object methods, however, have implementations that aren't particularly useful in general
- In most cases it is a good idea to override these methods with more useful versions.
- In other cases it is **required** if you want your objects to correctly work with other class libraries.

Some Object class methods

- Object methods of interest:
  - clone
  - equals
  - hashcode
  - toString
  - finalize
- Other object methods
  - getClass
  - wait, notify, notifyAll (relevant for threaded programming)

toString() method

- The Object method

  String toString();

  is intended to return a readable textual representation of the object upon which it is called. This is great for debugging!
- Best way to think of this is using a print statement. If we execute:

  System.out.println(someObject);

  we would like to see some meaningful info about someObject, such as values of iv's, etc.

default toString()

- By default toString() prints total garbage that no one is interested in

```
getClass().getName() + '@' + Integer.toHexString(hashCode())
```

- By convention, print simple formatted list of field names and values (or some important subset).
- The intent is not to overformat.
- Typically used for debugging.
- Always override toString()!

finalize() method

- Called as final step when Object is no longer used, just before garbage collection
- Object version does nothing
- Since java has automatic garbage collection, finalize() does not need to be overridden reclaim memory.
- Can be used to reclaim other resources – close streams, database connections, threads.
- However, it is strongly recommended *not* to rely on this for scarce resources.
- Be explicit and create own dispose method.