Inheritance:

- `extends` keyword to create a subclass
- `final` - prevents overriding both class and method
- `abstract` - can't be instantiated, only subclassed
- `interface` - can be implemented with `implements` by a class
  - multiple interfaces can be implemented
  - variables are public static by default
- can call old constructor with `super()` and function with `super.fnName()`
- **Polymorphism** allows to refer to objects of a subclass using a superclass reference e.g. `Animal myAnimal = new Dog();`. Static methods can't be overridden only hidden
- `instanceof` checks if an object is an instance of a class
- `getClass` returns the class of an object it can be used to compare classes `instance.getClass() == Dog.class`
- when comparing objects use `equals()` instead of `==` which compares references

Wrapper classes:

- Autoboxing: automatic conversion of primitive types to the object of their corresponding wrapper classes `Character ch = 'a';` and `char myChar = new Character('T');`
- Contain constants and useful functions

Useful functions:

```
// number conversion
Integer.toHexString(num) /*and*/ Integer.parseInt(String hex, int radix)
// strings
"Hello World!".split("o")   // splits string by "o" ["Hell", " W", "rld!"]
"Hello World!".substring(3/*begin index*/, 7 /*end index*/)
// arraylists
void add(int index, T obj) booean add(T obj) // returns true if the collaction changed
remove(int index) /*and*/ get(int index) // boath return the object they arge
int size() and bool isEmpty()
// Generics
class Box<T> { T var; void set(T obj) { var = obj; } T get() { return var; } }
// loops
for (int i = 1; i <= 5; i++) // do for 5 times
for (String name : names)    // do for names.length()
do {} while (condition)      // do while condition is true
```