# Abstract classes

An abstract class is a class declared abstract—it may or may not include abstract methods

Abstract classes cannot be instantiated, but they can be inherited

An ***abstract method*** is a method that is declared without an implementation (without braces, and followed by a semicolon)

If a class includes abstract methods, then the class itself must be declared abstract.

It can have constructors and static methods.

When an abstract class is subclassed, the subclass usually provides implementations for all of the abstract methods in its parent class. However, if it does not, then the subclass must also be declared abstract.

It can have final methods which will force the subclass not to change the body of the method.

## When to use abstract classes:

* You want to share code among several closely related classes.
* You expect that classes that extend your abstract class have many common methods or fields, or require access modifiers other than public (such as protected and private).

# String

* A sequence of characters

The String class is immutable, so that once it is created a String object cannot be changed. The String class has a number of methods, some of which will be discussed below, that appear to modify strings. Since strings are immutable, what these methods really do is create and return a new string that contains the result of the operation.

Methods:

* Length
* charAt
* concat
* equals
* equalsIgnoreCase
* isEmpty
* trim

# StringBuilder

StringBuilder objects are like String objects, except that they can be modified. Internally, these objects are treated like variable-length arrays that contain a sequence of characters. At any point, the length and content of the sequence can be changed through method invocations.

# Regular expressions (Regex)

is a sequence of characters that specifies a pattern which can be searched for in a text.

## Boundary matchers

Next, there’s syntax to specify the position of the matched sequence in the original text you’re searching. If you only need to filter out the strings that start with an email address or something, this is extremely useful.

^ The beginning of a line.

$ The end of a line.

\b A word boundary.

\B A non-word boundary.

\A The beginning of the input.

\G The end of the previous match.

\Z The end of the input but for the final terminator, if any.

\z The end of the input.

A noteworthy combination of the boundary matchers is the “^pattern$” which will only match the text if it is the full pattern.

## Logical operations

Now we’re getting into more advanced territory. If a pattern is more than a single character long, it will match a longer string too. In general “XY” in the regex syntax matches X followed by Y. However, there’s also an OR operation, denoted by the post “|”. The “X|Y” regex means it is either X or Y. This is a very powerful feature; you can combine the character classes or sequences of characters (include them in brackets).

## Quantifiers

On top of everything, you can say how many times the sequence of characters can be repeated for the match. The regex “1” only matches the input “1”, but if we need to match a string of any length consisting of the character “1” you need to use one of the following quantifiers.

\* matches zero or more occurrences.

+ matches one or more occurrences.

? matches zero or one occurrence.

## Groups and backreferences

A group is a captured subsequence of characters which may be used later in the expression with a backreference. We’ve mentioned already that if you enclose a group of characters in parentheses, you can apply quantifiers or logical or to the whole group. What is even more awesome is that you can refer to the actual characters in the text matched by the group, later.

Here’s how you do it:

(...) – defines a group, in the parantheses

\N – refers to a matched group

For example:

(\d\d) – a group of two digits

(\d\d)/\1– two digits repeated twice, \1 – refers to the matched group, so this regular expression matches the strings that look like “aabb” where a and b are both digits.

<https://www.freeformatter.com/java-regex-tester.html>