

COMMAND LINE PARSER

PROGRESS PRESENTATION

29 April 2019

CommandLineParser (92 tests) 37 failed	
CommandLineParserTests (28)	61 ms
CommandLineParser.Tests (28)	61 ms
ParserTests (21)	54 ms
ConflictingOptions	1 ms
DependentOptions	3 ms
GroupedOptionsLastWithParameter	2 ms
GroupedParameterLessOptions	1 ms
MandatoryOptionMissing	7 ms
MandatoryOptionMissingNoArgs	22 ms
MissingMandatoryOption	< 1 ms
MissingMandatoryParameter	< 1 ms
OptionsAfterDoubleMinusArePlainArguments	1 ms
OptionsHaveCorrectValues	2 ms
OptionsNotGivenHaveNullValue	< 1 ms
OptionsNotGivenNotParsed	< 1 ms
OptionsWithoutParametersHaveNullValue	< 1 ms
ParameterInDomainAccepted	1 ms
ParameterNotInDomainRefused	1 ms
ParameterOutOfBoundsRefused	< 1 ms
ParsedOptionContainsAllNames	2 ms
ParsedOptionHasCorrectValue	1 ms
TooFewPlainArgs	< 1 ms
TooManyPlainArgs	< 1 ms
WrongParameterTypeGiven	< 1 ms
SettingTests (7)	6 ms
DuplicatLongNames	< 1 ms
DuplicatShortNames	< 1 ms
EmptyProgramName	1 ms
HelpPrintsSomething	1 ms
MinPlainArgsLargerThanMaxPlainArgs	< 1 ms
NoNames	< 1 ms
NullProgramName	< 1 ms
CommandLineParserXUnitTests (64)	134 ms
CommandLineParserTests (64)	134 ms
IntParameterTests (7)	17 ms
ParseResultTests (17)	17 ms
ParsedOptionTests (4)	22 ms
ParserTests (14)	27 ms
ProgramSettingsTests (17)	46 ms
StringParameterTests (5)	5 ms

- MSTest Test Project and xUnit Test Project

CommandLineParser (92 tests) 37 failed	
CommandLineParserTests (28)	61 ms
CommandLineParser.Tests (28)	61 ms
ParserTests (21)	54 ms
SettingTests (7)	6 ms
CommandLineParserXUnitTests (64)	134 ms
CommandLineParserTests (64)	134 ms
IntParameterTests (7)	17 ms
ParseResultTests (17)	17 ms
ParsedOptionTests (4)	22 ms
ParserTests (14)	27 ms
ProgramSettingsTests (17)	46 ms
StringParameterTests (5)	5 ms

```

public void PrintHelp(System.IO.TextWriter writer)
{
    printHelpHeader(writer);
    writer.WriteLine();

    printGNUOptions(writer);
    writer.WriteLine();

    printGNUStandardOptions(writer);
    writer.WriteLine();

    //TODO documentation for plain arguments

    writer.Flush();
}

```

```

private void printHelpHeader(TextWriter writer)
{
    writer.WriteLine(indentSpace + programName + " [options]");
}

private void printGNUStandardOptions(TextWriter writer)
{
    writer.WriteLine(indentSpace + "GNU Standard Options");

    writer.WriteLine(indentSpace + indentSpace + "--help Print this help message.");
    writer.WriteLine(indentSpace + indentSpace + "-V, --version Print version information on standard output, then exit successfully.");
    writer.WriteLine(indentSpace + indentSpace + "--Terminate option list.");
}

private void printGNUOptions(TextWriter writer)
{
    writer.WriteLine(indentSpace + "GNU Options");

    foreach (Option opt in Options)
    {
        printOption(writer, opt);
    }
}

```

C:\Program Files\dotnet\dotnet.exe

time [options] command [arguments...]

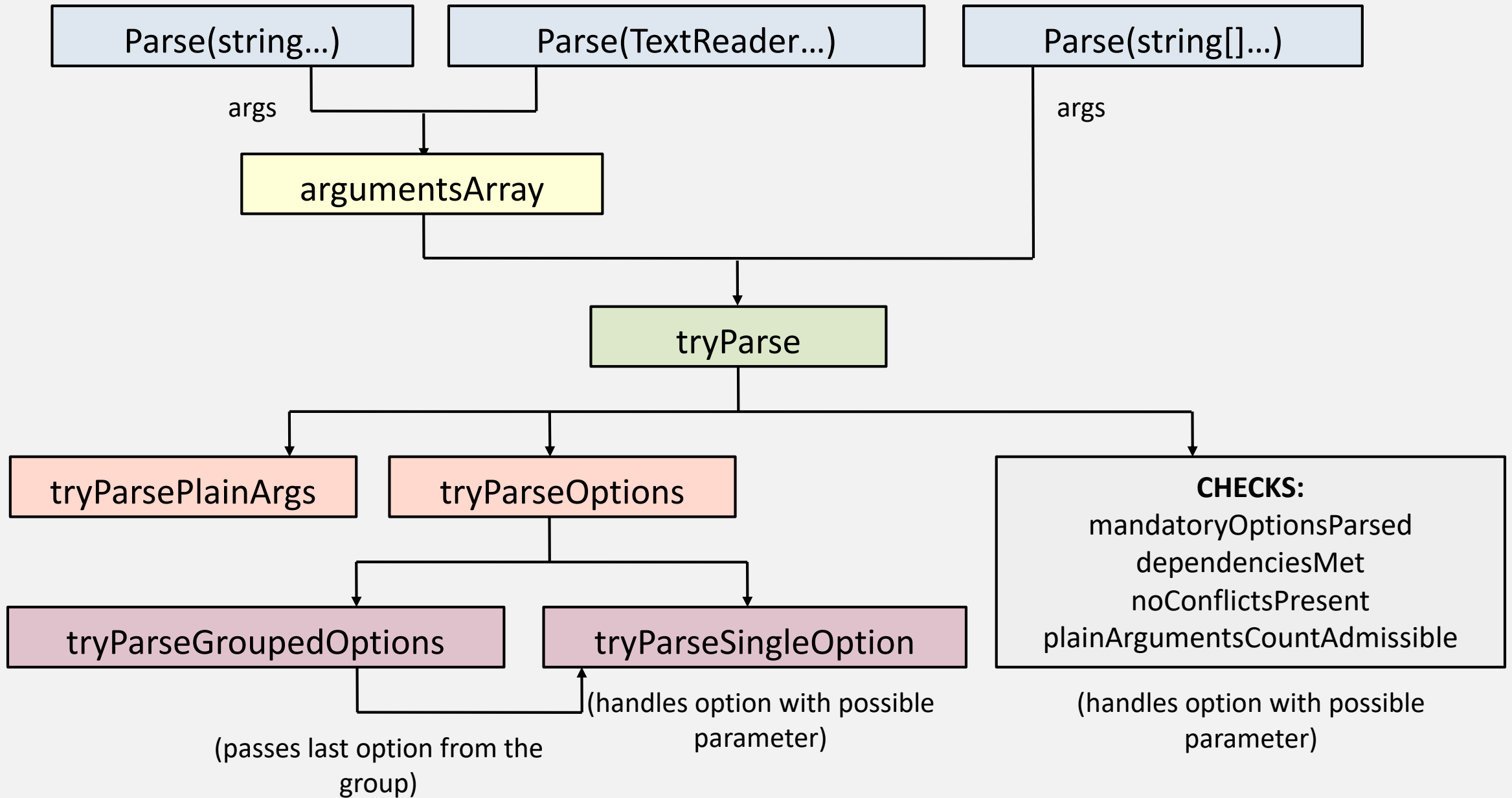
GNU Options

- f FORMAT, --format=FORMAT
Specify output format, possibly overriding the format specified in the environment.
- p, --portability
Use the portable output format.
- o FILE, --output=FILE
Do not send the results to stderr, but overwrite the specified file.
- a, --append
(Used together with -o.) Do not overwrite but append.
- v, --verbose
Give very verbose output about all the program knows about.
- V, --version
Print version information on standard output, then exit successfully.
- help
Print a usage message on standard output and exit successfully.

GNU Standard Options

- help Print a usage message on standard output and exit successfully.
- V, --version
Print version information on standard output, then exit successfully.
- Terminate option list.

```
internal List<Option> MandatoryOptions = new List<Option>();  
internal Dictionary<string, Option> OptionsDictionary = new Dictionary<string, Option>();  
internal Dictionary<Option, Option> OptionDependencies = new Dictionary<Option, Option>();  
internal List<List<Option>> OptionConflicts = new List<List<Option>>();
```



```
Regex singleOptionRegex = new Regex(@"^(\-{1}(?<optionName>[A-Za-z]{1})|\-{2}(?<optionName>[A-Za-z]{2,}))(\={1}(?<paramString>[0-9]+)|(?<paramString>[0-9]*))$");
```

```
var singleOptionMatch = singleOptionRegex.Match(args[argsIterator])
```

```
string optionName;
```

```
string paramString;
```

```
if (singleOptionMatch.Success) {
```

```
    optionName = singleOptionMatch.Groups["optionName"].ToString();
```

```
    paramString = singleOptionMatch.Groups["parameterString"].ToString();
```

```
}
```