Cron Token Parser Class Hierarchy

```
┌──────────────┐        ┌──────────────┐        ┌──────────────┐
│  <Validator> │        │  <Ordered>   │        │ <Evaluable>  │
└──────────────┘        └──────────────┘        └──────────────┘
```

AbstractCronToken ◆━ **CronTokenCategory**

Uses

| HourCronToken | HourCronToken | HourCronToken | HourCronToken | HourCronToken |

AbstractTokenValueResolver

| StepOnlyResolver | RangeStepResolver | FixedValueResolver | SpecialCharResolver |

| CronParsingService |

CronTokenFactory	◆━	CronTokenizer
getCronTokenInstances(): List<AbstractCronToken>

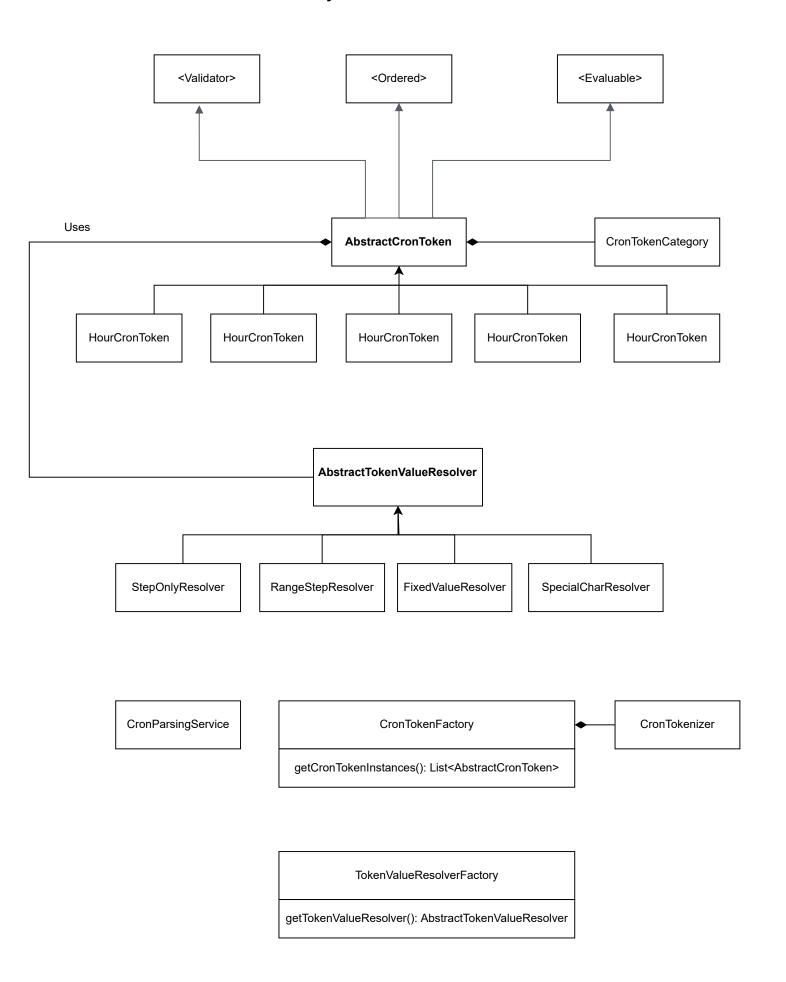| TokenValueResolverFactory |
| --- |
| getTokenValueResolver(): AbstractTokenValueResolver |

Process Description

| Tokenisation [Breaking Cron Expression into individual tokens | Mapping each token to corresponding Token Implementation | Choosing the Right Resolver for the tokenImpl. | Evaluate the token within the permissible values |

Example

Step 1. Input Cron Expression

```
*/15 0 1,15 * 1-5 /usr/bin/find
```

Step 2. Tokenize

```
Pair<CronTokenCategory, String>
Pair.of(CronTokenCategory.MINUTE, "*/15")
Pair.of(CronTokenCategory.HOUR, "0")
Pair.of(CronTokenCategory.DAY_OF_MONTH, "1,15")
Pair.of(CronTokenCategory.MONTH, "*")
Pair.of(CronTokenCategory.DAY_OF_WEEK, "1-5")
Pair.of(CronTokenCategory.COMMAND, "/usr/bin/find")
```

Step 3. Get Equivalent CronTokenType

```
list<AbstractCronToken>
[
new MinuteCronToken (value)
new HourCronToken (value)
new DayOfMonthCronToken (value)
new MonthCronToken (value)
new DayOfWeekCronToken (value)
new CommandCronToken (value)
]
```

Step4: For each value of CronTokenType find
        the correct resolver based on the token value,
        Apply the right `PermittedValue` Value to the resolver

```
if (value is of type */15 for token type MinuteCronToken then
CronTokenValue resolver will be StepOnlyResolver, With
PermittedValue as 0 to 60.
```

Step5: Resolve the applicable value based on
        the token type and permitted value

```
return 0 15 30 45 as list
```