Use `#` as indicator for BibTeX string constants

Bibtex supports string constants. The entry editor should support that, too. Affected code is (at least) `org.jabref.logic.importer.fileformat.BibtexParser#parseFieldContent` and `org.jabref.logic.bibtex.FieldWriter#formatAndResolveStrings`

Decision Drivers

- Full BibTeX compatibility
- Intuitive for the user
- UI backwards compatible

Considered Options

- Wrap string constants in `#`
- Replace the internal `#` in JabRef by the non-used character `§`
- Replace the internal `#` in JabRef by the non-used character `%`
- Replace the internal `#` in JabRef by the non-used character `&`
- Replace the internal `#` in JabRef by a single `&`
- Change the data type of a field value
- Wrap plan # in curly braces: `{#}`

Decision Outcome

Chosen option: “Wrap string constants in #”, because already existing behavior.

Pros and Cons of the Options

Wrap string constants in `#`

When BibTeX on the disk contains `field = value`, it is rendered as `field = #value#` in the entry editor. The user then knows that `value` is a BibTeX string.

Example:

left: BibTeX; right: Entry Editor

```
@String{ value = "example" }

field1 = value     -> #value#
field2 = {value}   -> value

field1 = value # value --> #value# # #value#
```

- Good, because In place in JabRef since more than 10 years
- Bad, because `#` is used in BibTeX for string concatenation
- Bad, because `#` is used in Markdown to indicate headings. When using Markdown in fields such as comments, this causes writing errors
- Bad, because `#` could appear in URLs

Replace the internal `#` in JabRef by the non-used character `§`

When BibTeX on the disk contains `field = value`, it is rendered as `field = §value§` in the entry editor. The user then knows that `value` is a BibTeX string.

Example:

left: BibTeX; right: Entry Editor

```
field1 = value     -> §value§
field2 = {value}   -> value

field1 = value # value --> §value§ # §value§
```

- Good, because easy to implement
- Good, because no conflict with BibTeX's `#` sign
- Bad, because documentation needs to be updated
- Bad, because `§` is not available on a US keyboard

Replace the internal `#` in JabRef by the non-used character `%`

Similar to "Replace the internal # in JabRef by the non-used character §"

Example:

left: BibTeX; right: Entry Editor

```
field1 = value     -> %value%
field2 = {value}   -> value

field1 = value # value --> %value% # %value%
```

- Good, because a user does not write LaTeX comments inside a string
- Bad, because `%` could be used in Markdown, too
- Bad, because `%` is used for comments in LaTeX and thus migtht cause confusion
- Bad, because `%` could appear in URLs: There is url percent encoding

Replace the internal `#` in JabRef by the non-used character `&`

Similar to “Replace the internal `#` in JabRef by the non-used character `§`”

Example:

left: BibTeX; right: Entry Editor

```
field1 = value     -> &value&
field2 = {value}   -> value

field1 = value # value --> &value& # &value&
```

- Good, because Users do not write LaTeX tables
- Bad, because `&` is a LaTeX command for tables

Replace the internal `#` in JabRef by a single `&`

Prefix strings with `&`

Example:

left: BibTeX; right: Entry Editor

```
field1 = value     -> &value
field2 = {value}   -> value

field1 = value # value --> &value # &value
```

- Good, because near to C syntax
- Bad, because `&` is a LaTeX command for tables
- Bad, because `&` could appear in URLs

Change the data type of a field value

`org.jabref.model.entry.BibEntry#setField(org.jabref.model.entry.field.Field, java.lang.String)` changes to `org.jabref.model.entry.BibEntry#setField(org.jabref.model.entry.field.Field, org.jabref.model.entry.field.Value)` (org.jabref.model.entry.BibEntry#setField(org.jabref.model.entry.field.Field, java.lang.String)). This would bring JabRef’s internal data model even more close to BibTeX

- Good, because more object-orientated design of BibTeX field storage

- Good, because eases implementation of an advanced field editor
- Bad, because high effort to implement