Karg: keyword arguments for Java

Keyword arguments in other languages

Most civilised programming languages (and some quite uncivilised ones) support some way of identifying the arguments passed to a function by name rather than by order, e.g.

```
person = directory.findThePerson(forename='Arthur', surname='Putey')
```

rather than

```
person = directory.findThePerson('Arthur', 'Putey')
```

This becomes especially useful when (as in the case of searching a directory for a person matching some criteria) not all of the arguments are compulsory, or some may have sensible default values:

```
# Finds Arthur Putey and Arthur Daley
arthurs = directory.findThePeople(forename='Arthur')

# Finds Jack Smith and Wolfie Smith
smiths = directory.findThePeople(surname='Smith')
```

A common convention is to specify the default value for a named argument in the signature of the function itself:

```
def pythonQuotation(quotation, freshness='stale'):
    print 'For your amusement, the following %s Python quotation:' % freshness
    print quotation

pythonQuotation('This is an ex-Parrot!')
pythonQuotation('Run away! Run away!', freshness='ever-fragrant')
```

Named arguments (or keyword arguments, as they're sometimes known) are especially useful when you want to specify the desired properties of some object to a builder of objects of that kind:

```
myPony = daddy.iWantAPony(colour='pink',
                          mane='lustrous',
                          magicPowers=['glitter stamp', 'rainbow yawn'])
```

Alas, Java has yet to attain this level of civilisation. Karg is a makeshift solution, lashed together with twigs and vine.

A simple example

A library can't modify Java's syntax, so Karg's solution involves some abuse of existing syntax to achieve a reasonably fluent alternative. Keywords are defined as constants:

```
public final class Quotation {
    public static final Keyword<String> FRESHNESS = Keyword.newKeyword();
}
```

which may be statically imported

```
import static org.example.keywords.Quotation.FRESHNESS;
```

and then used as follows:

```
pythonQuotation("Welease Wodewick!", FRESHNESS.of("somewhat gamey"));
```

If you don't like static imports, or want to disambiguate keywords with the same names but different meanings or intended contexts, consider scoping keywords to their containing classes, e.g.

```
import org.example.keywords.Quotation;
```

and then

```
pythonQuotation("The larch", Quotation.FRESHNESS.of("distinctly mildewed"));
```

A method that will receive keyword arguments needs to do the following:

```
public void pythonQuotation(String quotation, KeywordArgument...argArray) {
    KeywordArguments args = KeywordArguments.of(argArray);
    String freshness = FRESHNESS.from(args, "stale");
    System.out.println(String.format(
        "For your amusement, the following %s Python quotation:", freshness
    ));
    System.out.println(quotation);
}
```

Note that a default value is given when we try to retrieve the value of a KeywordArgument from the KeywordArguments collection.

A less simple example

As mentioned earlier, keyword arguments are especially handy when you want to tell a builder of some kind about the desired properties of the object you want it to build:

```
Pony myPony = daddy.iWantAPony(COLOUR.of("pink"),
                               MANE.of("lustrous"),
                               MAGIC_POWERS.of(newArrayList(
                                   "glitter stomp",
                                   "rainbow yawn")));
```

Sometimes it's useful to be able to treat this bundle of properties as an object in its own right:

```
KeywordArguments defaultPonySpec = KeywordArguments.of(
    COLOUR.of("grey"),
    MANE.of("lank"),
    MAGIC_POWERS.of(emptyList()));

KeywordArguments happyPonySpec = defaultPonySpec.with(
    MANE.of("lustrous"));

KeywordArguments pinkPonySpec = happyPonySpec.with(
    COLOUR.of("pink"));

KeywordArguments magicPonySpec = pinkPonySpec.with(
    MAGIC_POWERS.of(newArrayList("glitter stomp", "rainbow yawn")));

Pony myPony = daddy.iWantAPony(magicPonySpec);
```

For this to work, the iWantAPony method must be overloaded to support being called with either varargs or a KeywordArguments object:

```
public Pony iWantAPony(KeywordArgument...argArray) {
    return iWantAPony(KeywordArguments.of(argArray));
}

public Pony iWantAPony(KeywordArguments args) {
    Pony newPony = new PonyImpl();
    newPony.setColour(COLOUR.from(args));
    newPony.setMane(MANE.from(args));
    newPony.setMagicPowers(MAGIC_POWERS.from(args));
    return newPony;
}
```

Getting Karg

You can get Karg from GitHub, or from Maven Central.