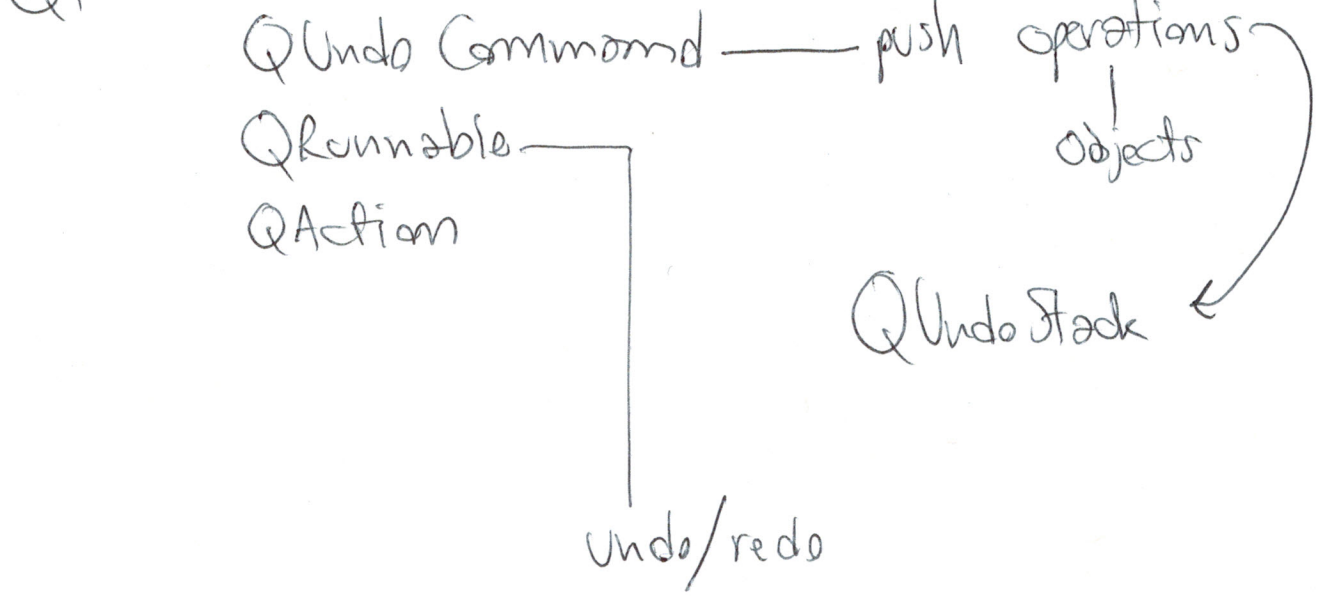
SEC 10.5

THE COMMAND PATTERN [Gamma 95]

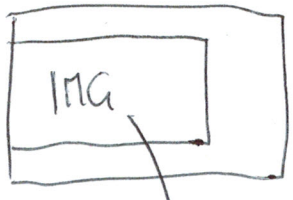
encapsulate operations as objects

EXAMPLE UNDO / REDO

Qt



10_11 to 10_14



MIRROR is an OBJECT

ADJUST is an OBJECT

① CREATE a new obj MIRROR ( )

② MIRROR::mirror() taking a copy in m_Saved

③ SAVE THE OBJ in QStack Undo

SEC 10.6

INTERNATIONALIZATION

```
tr("My string");
```

```
QObject::tr()
```

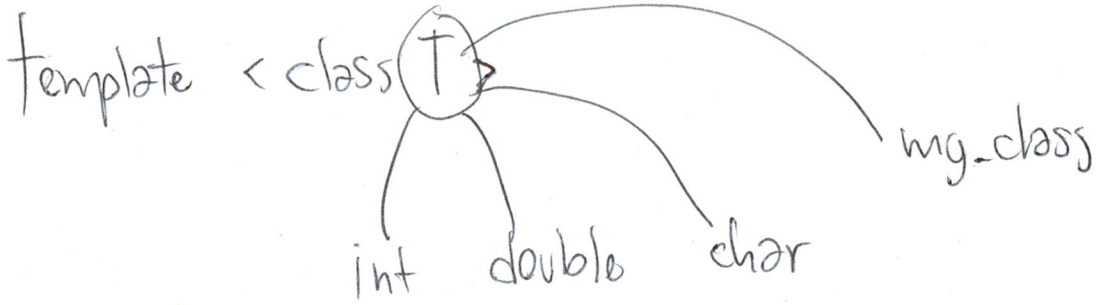
TOOLS IN THE Qt FRAMEWORK	lupdate	linguist	lrelease

CHAP 11

TEMPLATES

template < ... >

↑ template parameters



< class T, class M >

< class T, class N, int M >

type parameters

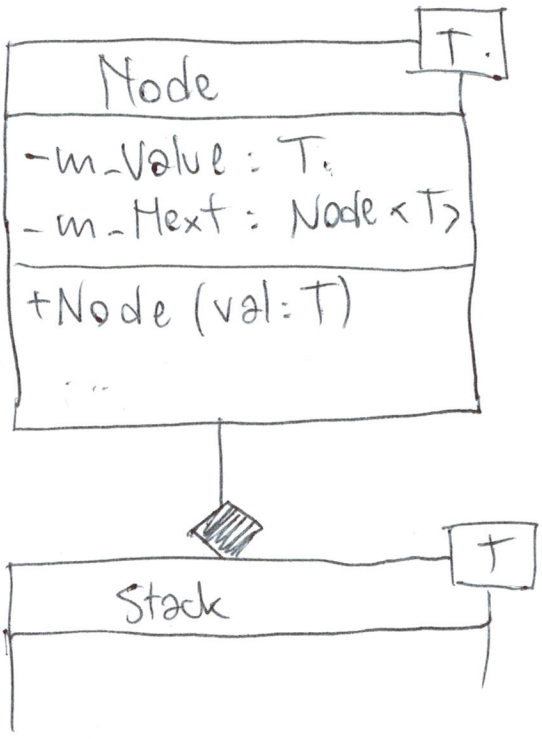
const parameter

```
template <class T>
class my-class {


}
```

```
template <class T, int max>
Buffor { T v[max]; }
```

SEC 11.1.1 FUNCTION TEMPLATE

SEC 11.1.2 CLASS TEMPLATES

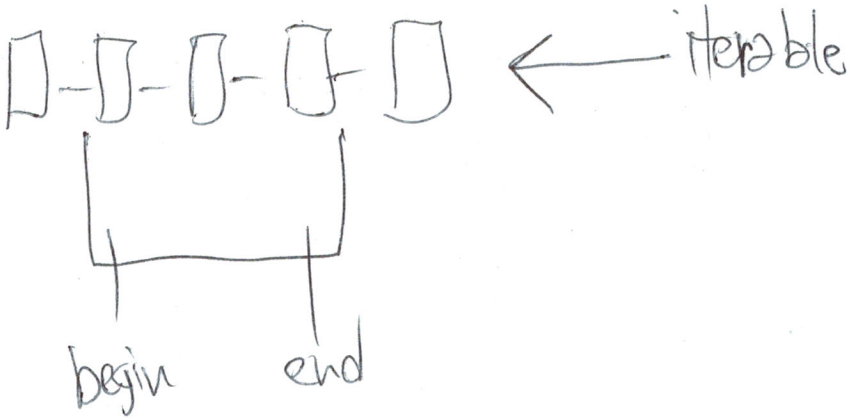


SEC 11.2

qSort

QtAlgorithms

qSort (RandomAccessIterator begin,
RandomAccessIterator end);



C++ overloading of operators

operator==

↓

COMPARE

SEC 11.3

QMap<..., ...>

textbookMap → QMap<QString, int>

deleteAll()
clear()

SEC 11.4 FUNCTION FUNCTORS

C FUNCTION POINTER
|
CALLBACK

```
QString name(){ return QString("Antonio"); }
QString name_1(){ return QString("Alan"); }
typedef QString (*funcPtr)();
...
        funPtr ptr = name; // ptr = name_1;
        QString v = ptr();
```

FUNCTOR (example)

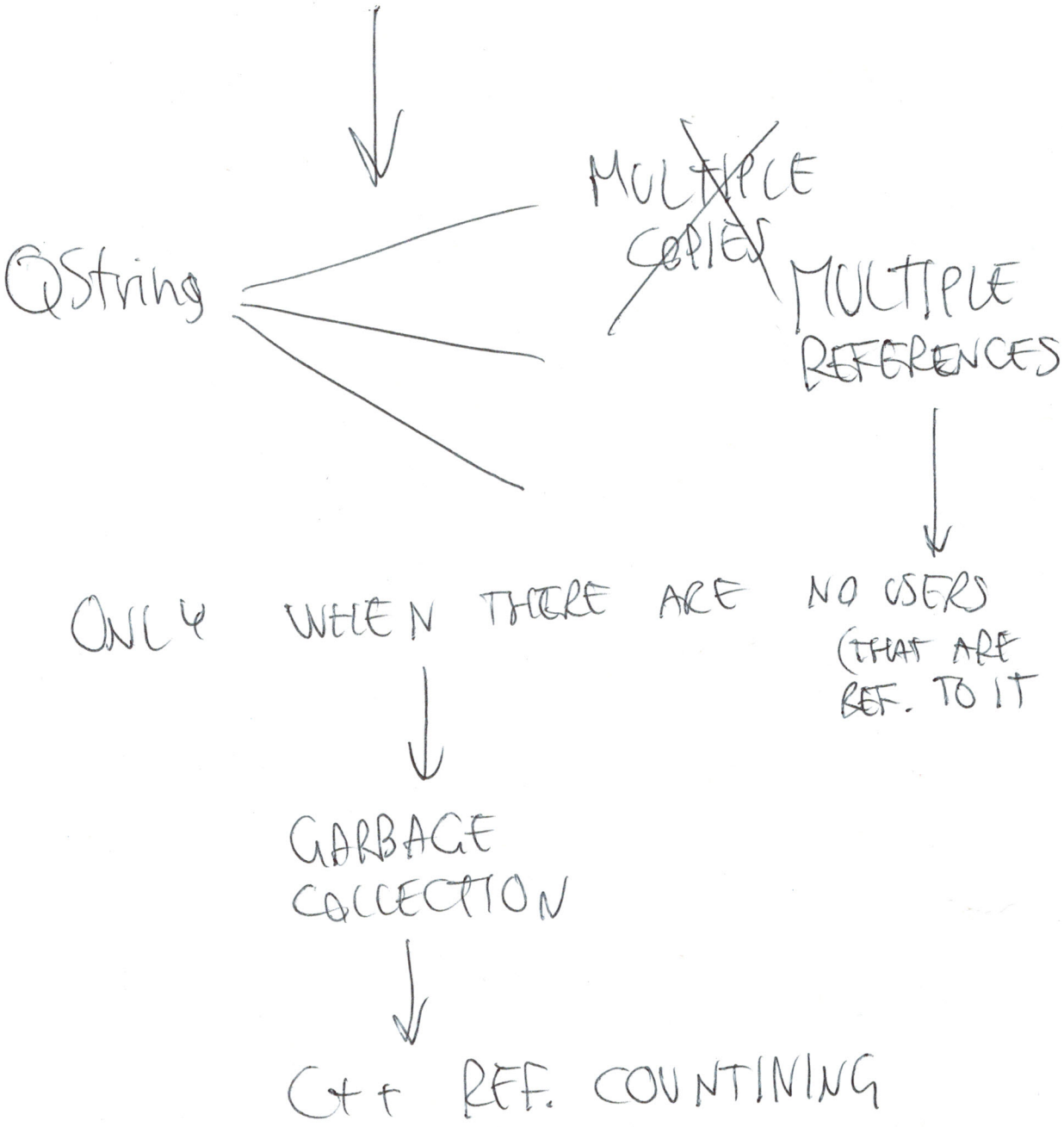
```
class name {
public:
    QString operator()() {
        return QString("Antonio");
    }
};

...

name ptr;
QString v = ptr()
```

SEC 11.5 FLYWEIGHT PATTERN

→ DON'T DO MULTIPLE COPIES OF A SINGLE DATA BUT KEEP ONLY ONE COPY

→ COPY on WRITE

USED in Qt → QString
→ QStringList
→ QVariant


QString
MULTIPLE COPIES
MULTIPLE REFERENCES
DELETE ONLY WHEN THERE ARE NO USERS (THAT ARE REF. TO IT
GARBAGE COLLECTION
C++ REF. COUNTING