CHAP 19 TYPES and EXPRESSIONS

SEC 19.7 EXPLICIT CONVERSIONS

CAST

(type) expr

EXAMPLE

```
double d = 3.14;
int i = (int) d;
```

Type + = Type (arg list);

EXAMPLE

```
double d = 3.14;
Complex c = Complex(d);
```

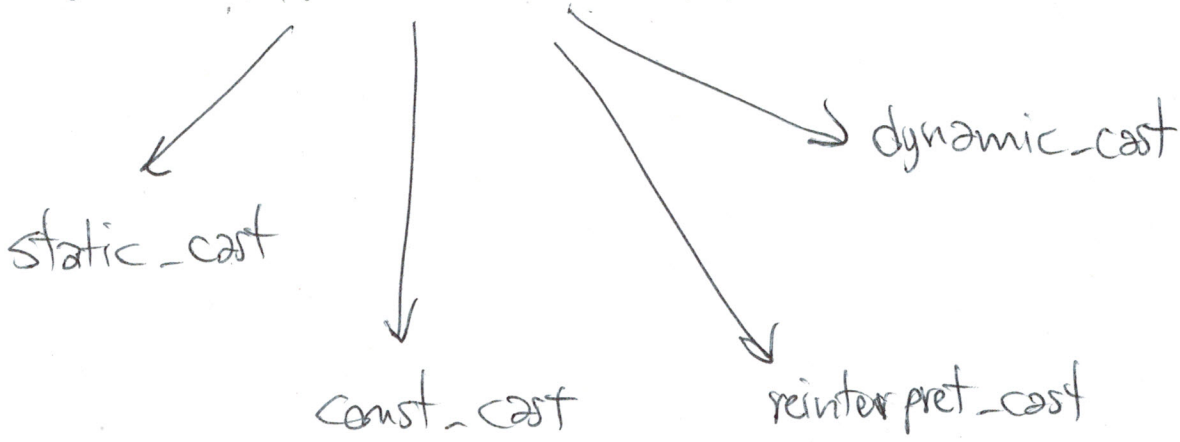
↓

```
Complex::Complex (int d){

}
```

Constructor-style CONVERSION

SEC 19.8 TYPECASTING ANSI C++ STD



SEC 19.8.1

static_cast<DestType>(expr)

→ BETWEEN RELATED TYPES
NUMBERS

COMPILE TIME CASTING

EXAMPLE

```
int i=3
double d=static_cast<double>(i);
```

SEC 19.8.1

```
const_cast<DestType>(expr);
```

REMOVES THE FACT THAT A VAR IS CONST

EXAMPLE

```
const int N=22;
int * pN = const_cast<int*>(&N);
*pN = 33;
```

→ NOW YOU CAN MODIFY N!

SEC 19.8.2

```
reinterpret_cast<DestType>(expr);
```

TO CONVERT BETWEEN UNRELATED TYPES (e.g. number and pointer)

```
Spam spam;
Egg* eggP;
eggP = reinterpret_cast<Egg*>(&spam);
eggP->scramble();
```

SEC 19.10

RTTI

RunTime
Type
Identification System

— C++

```
dynamic_cast<DataType>(expr)
typeid(expr)
```


```
B* ptr;
dynamic_cast<D*>(ptr)
```

IF D==B
OR D→B

UPCAST

EQUIVALENT TO STATIC CAST

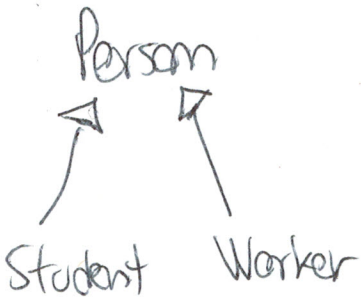
DOWN CAST

IF FAILS VALUE RETURNED IS NULL

SEC 19.10.1 typeid OPERATOR

USAGE check if two object instances are from the same class

EXAMPLE



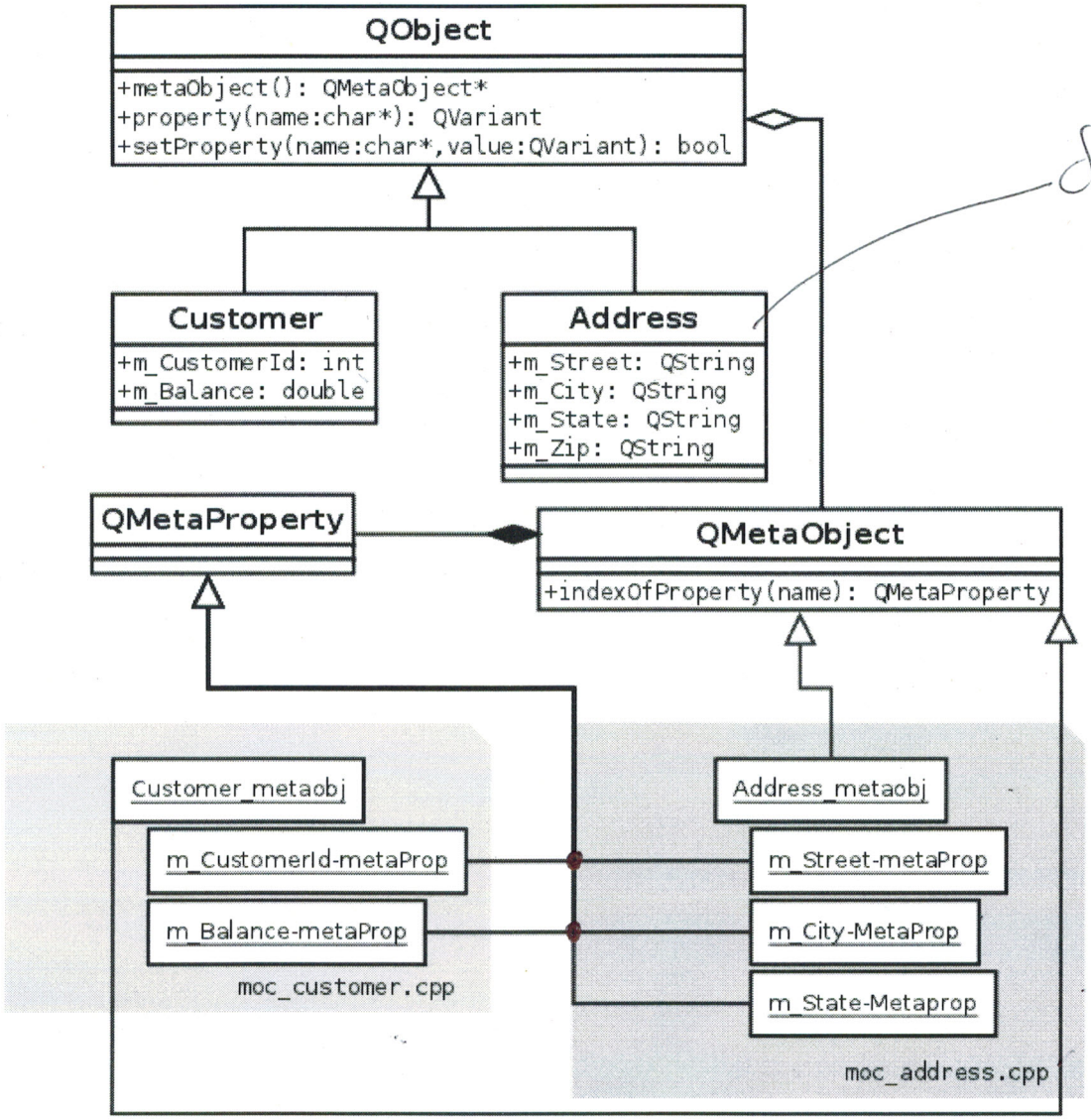

```
void f (Person& pRef){

    if (typeid(pRef) == typeid(Student)){
        // its a student
    } else
        // something else

}
}
```

INSTANCES

TYPES



class Address : public QObject
Q_OBJECT

FIGURE 12.1 Ezust p379

(Copyright Ezust)

CHAP 12

Qt objects and types supports REFLECTION

SELF EXAMINATE AN OBJECT FOR ITS MEMBERS

SEC 12.1 QMetaObject —— MetaObject Pattern

META OBJECT DESCRIBES THE STRUCTURE OF AN OBJECT

SEC 12.2

qobject_cast<DestType>(expr)

RUNTIME

ON FAILURE RETURNS ∅

adhere to ANSI C++ (but is not standard)

QObject::inherits(" ")

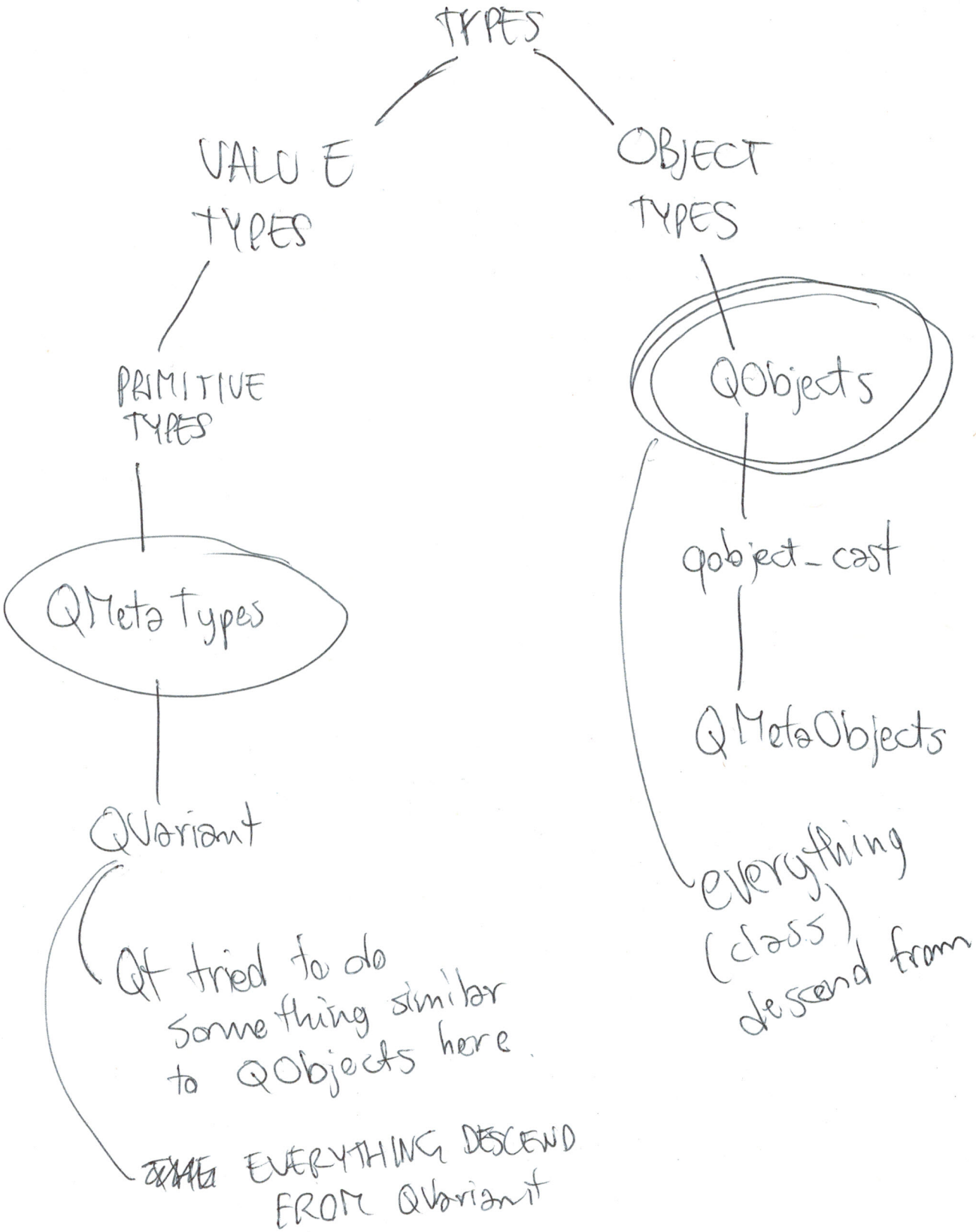
"className"

QObject*

DestType* qobject<DestType*>(QObject* qoptr)

SEC 12.6

REM


TYPES
VALUE TYPES
OBJECT TYPES
PRIMITIVE TYPES
QObjects
QMetaTypes
qobject_cast
QMetaObjects
QVariant
everything (class) descend from
Qt tried to do something similar to QObjects here.
EVERYTHING DESCEND FROM QVariant

① AFTER YOUR CLASS DECLARATION

```
Q_DECLARE_METATYPE(myclass);
```

②

```
qRegisterMetaType<T>();
```

OR

```
qRegisterMetaType<T>("MyClass");
```

CHAP 13

MODELS and VIEWS

SEPARATE IN DIFFERENT CLASSES

- MODEL
  - data
- VIEW
  - presentation for the user

WHY?

- reduce code complexity
- improve maintenance
- to easily maintain several views of data

SEC 13.1

MODEL - VIEW - CONTROLLER (MVC)

[Gamma95] ——— MODEL application object

VIEW screen presentation

CONTROLLER defines the way the user interface react to user input

---

CONTROLLER ——— manages interactions between components

FACTORY METHODS (TODO)

DELEGATES

CREATION and DESTRUCTION code

A delegate can for example control the rendering and editing of individual items in views.