Constructors

- Are special member functions for instantiation of an object.
- Will be automatically called or can be triggered by a `new` or `new[]`.
- Have the same name as the class and return nothing.
- Can be defined inside or outside the class.
- Are typically overloaded to instantiate an object in different ways.

Default

- Default constructor
  - Needs no argument. ⇨ Can have default values for each parameter
  - Can be called in two variations.

```
Account account;
Account* p = new Account;
```

  - Will be automatically generated by the constructor if possible.
  - Invokes automatically all constructors of the base classes and the attributes.

The compiler needs the default-constructor to automatically create instances.

Copy

- Copy constructor
  - Expects a constant lvalue reference to an instance of the class.

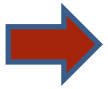
```
class Account{
public:
    Account(const Account& other);
};
```

  - Copies all arguments.
  - The Object and `other` has afterwards the same value.

Move

- Move constructor
  - Expects a non-constant rvalue reference to an instance of the class

```
class Account{
public:
    Account(Account&& other);
};
```

  - Moves all arguments
  - `other` is afterwards in a moved-from state  to reuse it you have to initialize it

Explicit Constructor

- Explicit declared constructors can not be used implicit.

```
class Account{
public:
    explicit Account(double b): balance(b){}
    Account (double b, std::string c): balance(b), cur(c){}
private:
    double balance;
    std::string cur;
};
Account account0 = 100.0;            // ERROR: implicit conversion
Account account1(100.0);             // OK: explicit invocation
Account account2 = {100.0, "EUR"};   // OK: implicit conversion
```

constructorExplicit.cpp

Constructor Delegation

- A constructor can call a constructor of the same class.
- This constructor must be called in the class initializer.

```
struct Account{
    Account(): Account(0.0){}
    Account (double b): balance(b){}
};
```

- Rules:
  - In case the first constructor is done, the object is generated.
  - Constructors cannot be invoked recursively.

  ➡ undefined behavior

- Idea:
  - Shared initialization tasks can be implemented in one constructor, and be used from all others.

constructorInitializer.cpp
constructorDelegation.cpp