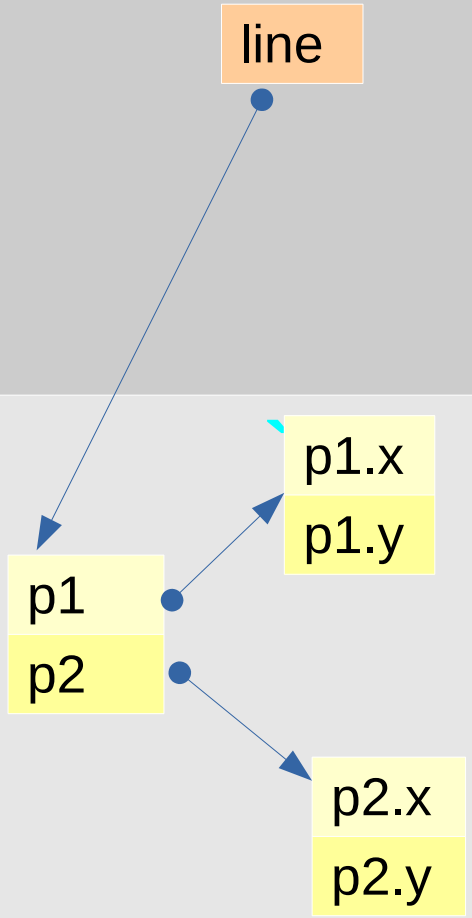
Composition and initialization

- Version 1: Dr. Ofir Pele
- Version 2: Dr. Miri Ben-Nissan
- Version 3: Dr. Erel Segal-Halevi

Composition *(folder 7)*

```
class Line {
   Point p1, p2;
};
```

C++	Java
Members are **objects**	Members are **pointers**

	C++	Java
Stack:	line.p1.x line.p1.y line.p2.x line.p2.y	
Heap:		


```
int main () {   Line line;   };
```

Construction

```
int main() {
   B b;
}
```

```
class A {
public:
 A() {
  std::cout <<
  "A - " <<
  "parameterless" <<
  " ctor\n";
 }
};
```

```
class B {
   A _a1,_a2;
public:
   B() {
      std::cout <<
      "B - parameterless " <<
      "ctor\n";
   }
};
```

Question: In what order are the constructors called?

Construction order

```
int main() {
   B b;
}
```

```
class A {
public:
 A() {
  std::cout <<
  "A - " <<
  "parameterless" <<
  " ctor\n";
 }
};
```

```
class B {
   A _a1,_a2;
public:
   B() {
      std::cout <<
      "B - parameterless " <<
      "ctor\n";
   }
};
```

```
// Answer: small to large
A - parameterless ctor
A - parameterless ctor
B - parameterless ctor
```

Destruction

```
int main() {
    B b;
}
```

```
class A {
public:
 ~A() {
  std::cout <<
  "A - " <<
  " dtor\n";
 }
};
```

```
class B {
    A _a1,_a2;
public:
    ~B() {
      std::cout <<
      "B - dtor\n";
    }
};
```

Question: In what order are the destructors called?

Destruction order

```
int main() {
   B b;
}
```

```
class A {
public:
 ~A() {
  std::cout <<
  "A - " <<
  " dtor\n";
 }
};
```

```
class B {
   A _a1,_a2;
public:
   ~B() {
     std::cout <<
     "B - dtor\n";
   }
};
```

```
// Answer: large to small
B - dtor
A - dtor
A - dtor
```

The paramaterless ctor (aka **default ctor**)

```
int main() {
   B b;
}
```

```
class A {
public:
  A(int a) {
    std::cout <<
    "A ctor with one
     parameter\n";
  }
};
```

```
class B {
   A _a1,_a2;
public:
   B() {
     std::cout <<
     "B – parameterless " <<
     "ctor\n";
   }
};
```

```
// compilation error
No parameterless ctor for _a1, _a2
```

The initialization list

```
int main() {
    B b(2,3);
}
```

```
class A {
public:
    A(int a) {
        std::cout <<
        "A (" << a << ") "

        << std::endl;
    }
};
```

```
class B {
    A _a1,_a2;
public:
    B(int i, int j)
    :_a1 (i), _a2(j)
    {
        std::cout
        << "B cons"
        << std::endl;
    }
};
```

```
// output
A (2)
A (3)
B cons
```

Initialization using pointers (1)

```
int main() {
   B b(2);
}
```

```
class A {
public:
   A(int a) {
      std::cout <<
      "A (" << a << ") "

      << std::endl;
   }
};
```

```
class B {
   A *_ap;
public:
   B(int i);
};

B::B(int i) {
   _ap = new A (i);
   cout << "B cons\n";
}
```

```
// output
A (2)
B cons
```

Initialization using pointers (2)

```
int main() {
    B b(2);
}
```

```
class A {
public:
    A(int a) {
        std::cout <<
        "A (" << a << ") "

        << std::endl;
    }
};
```

```
class B {
    A *_ap;
public:
    B(int i);
};

B::B(int i)
    : _ap (new A(i))
{
    cout << "B cons\n";
}
```

```
output //
A (2)
B cons
```

The initialization list

```
int main() {
    B b(2,3);
}
```

```
class A {
public:
    A(int a) {
        std::cout <<
        "A (" << a << ") "

        << std::endl;
    }
};
```

```
class B {
    A _a1,_a2;
public:
    B(int i, int j)
    :_a1 (i), _a2(j)
    {
        std::cout
        << "B cons"
        << std::endl;
    }
};
```

```
// output
A (2)
A (3)
B cons
```

The initialization list

1. Initialization of object members.
2. Initialization of constants and reference variables.
3. Initialization of parent class.
4. It is faster and safer to use the initialization list than initialization in the constructor