Achtung gemein! Klassen

```
class N{
int x; int y;
public:
  N (int X, int Y): x(X), y(Y) {};

  N (const N& n): N(n.y, n.x) {};

  std::ostream& print(std::ostream& out) const {
    return out << x << y;
  }
};

std::ostream& operator<<(std::ostream& o, N n){
  return n.print(o);
}

int main(){
  N n(1,2);
  const N m(n);
  std::cout << n << m << std::endl;
}
```

Was gibt `main` aus?

1. 1212
2. 2121
3. 1221
4. 2112
5. kompiliert gar nicht.

Achtung gemein! Klassen

```
class N{
int x; int y;
public:
  N (int X, int Y): x(X), y(Y) {};

  N (const N& n): N(n.y, n.x) {};

  std::ostream& print(std::ostream& out) const {
    return out << x << y;
  }
};

std::ostream& operator<<(std::ostream& o, N n){
  return n.print(o);
}

int main(){
  N n(1,2);
  const N m(n);
  std::cout << n << m << std::endl;
}
```

Was gibt `main` aus?

1. 1212
2. 2121
3. 1221
4. 2112
5. kompiliert gar nicht.

Achtung gemein! Klassen

```
class N{
int x; int y;
public:
  N (int X, int Y): x(X), y(Y) {};

  N (const N& n): N(n.y, n.x) {};

  std::ostream& print(std::ostream& out) const {
    return out << x << y;
  }
};

std::ostream& operator<<(std::ostream& o, N n){
  return n.print(o);
}

int main(){
  N n(1,2);
  const N m(n);
  std::cout << n << m << std::endl;
}
```

Was gibt `main` aus?

1. 1212
2. 2121
3. 1221
4. 2112 ●
5. kompiliert gar nicht.

Achtung gemein! Klassen

```
class N{
int x; int y;
public:
  N (int X, int Y): x(X), y(Y) {};

  N (const N& n): N(n.y, n.x) {};

  std::ostream& print(std::ostream& out) const {
    return out << x << y;
  }
};

std::ostream& operator<<(std::ostream& o, N n){
  return n.print(o);
}

int main(){
  N n(1,2);
  const N m(n);
  std::cout << n << m << std
}
```

Kopierkonstruktor

pass by value!

der formale Parameter n wird mit dem Argument initialisiert, dabei wird der Kopierkonstruktor aufgerufen

Was gibt `main` aus?

1. 1212
2. 2121
3. 1221
4. 2112
5. kompiliert gar nicht.