



Compilers

Java Interfaces

Specify relationships between classes without inheritance

```
interface PointInterface { void move(int dx, int dy); }
```

```
class Point implements PointInterface {  
    void move(int dx, int dy) { ... }  
}
```

“Java programs can use interfaces to make it unnecessary for related classes to share a common abstract superclass or to add methods to Object.”

In other words, interfaces play the same role as multiple inheritance in C++, because classes can implement multiple interfaces

class X implements A, B, C { ... }

- A graduate student may be both an University employee and a student

class GraduateStudent implements Employee, Student
{ ... }

- No good way to incorporate Employee, Student methods for grad students with single inheritance

Methods in classes implementing interfaces need not be at fixed offsets.

```
interface PointInterface { void move(int dx, int dy); }
```

```
class Point implements PointInterface {
```

```
    void move(int dx, int dy) { ... } }
```

```
class Point2 implements PointInterface {
```

```
    void dummy() { ... }
```

```
    void move(int dx, int dy) { ... } }
```

- Dispatches `e.f(...)` where `e` has an interface type are more complex than usual
 - Because methods don't live at fixed offsets
- One approach:
 - Each class implementing an interface has a lookup table `method names → methods`
 - Hash method names for faster lookup
 - hashes computed at compile time