Compilation

0368-3133 2016/17a

Lecture 11b

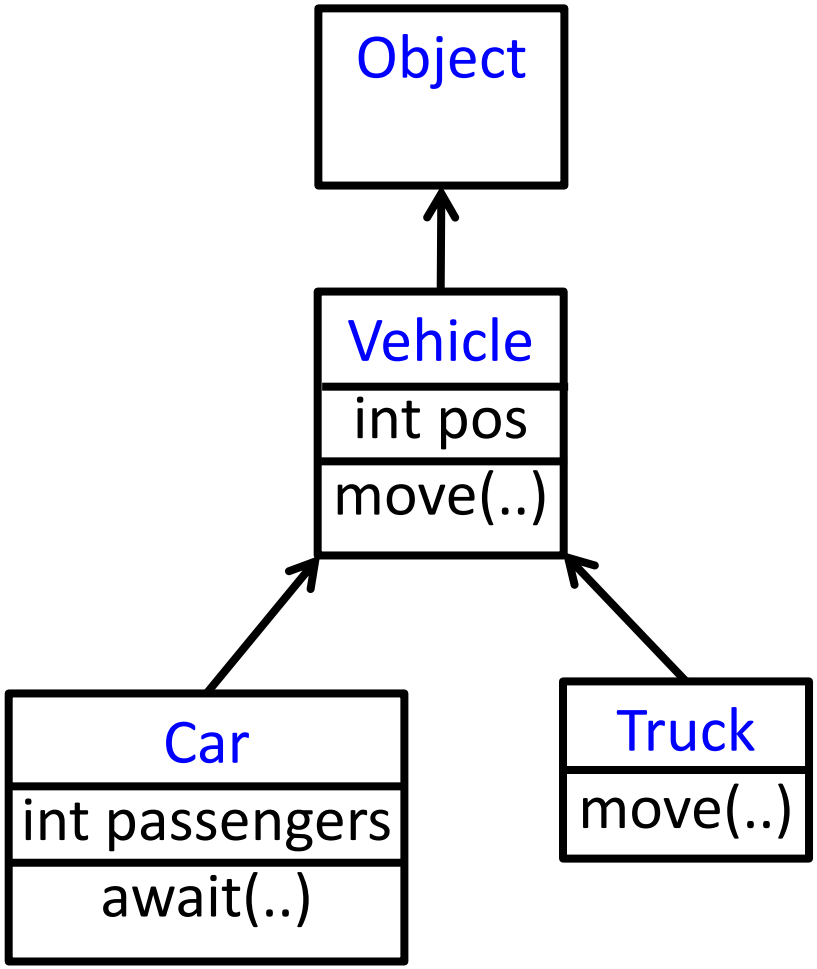
Compiling Object-Oriented Programs

Noam Rinetzky

Object Oriented Programs

- C++, Java, C#, Python, …
- Main abstraction: **Objects** (usually of type called class)
  - Code
  - Data
- Naturally supports **A**bstract **D**ata **T**ype implementations
- Information hiding
- Evolution & reusability
- Important characteristic: Extension/Inheritance

A Simple Example



A Simple Example

```
class Vehicle extends Object {
  int pos = 10;
  void move(int x) {
    pos = pos + x ;
  }
}

class Truck extends Vehicle {
  void move(int x){
    if (x < 55)
      pos = pos + x;
  }
}

class Car extends Vehicle {
  int passengers = 0;
  void await(vehicle v){
    if (v.pos < pos)
      v.move(pos - v.pos);
    else
      this.move(10);
  }
}
```

```
class main extends Object {
  void main() {
    Truck t = new Truck();
    Car c = new Car();
    Vehicle o = c;
    o.move(60);
    t.move(70);
    c.await(t);
  }
}
```

A Simple Example

```
class Vehicle extends object {
  int pos = 10;
  void move(int x) {
    position = position + x ;
  }
}

class Truck extends Vehicle {
  void move(int x){
    if (x < 55)
      pos = pos + x;
  }
}

class Car extends Vehicle {
  int passengers = 0;
  void await(vehicle v){
    if (v.pos < pos)
      v.move(pos - v.pos);
    else
      this.move(10);
  }
}
```

```
class main extends Object {
  void main() {
    Truck t = new Truck();
    Car c = new Car();
    Vehicle o = c;
    o.move(60);
    t.move(70);
    c.await(t);
  }
}
```

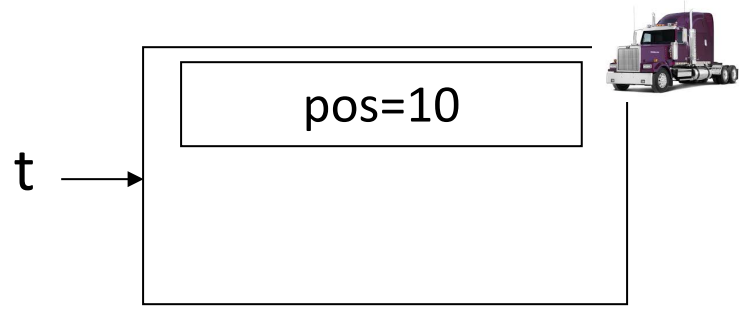
A Simple Example

```
class Vehicle extends object {
  int pos = 10;
  void move(int x) {
    position = position + x ;
  }
}

class Truck extends Vehicle {
  void move(int x){
    if (x < 55)
      pos = pos + x;
  }
}

class Car extends Vehicle {
  int passengers = 0;
  void await(vehicle v){
    if (v.pos < pos)
      v.move(pos - v.pos);
    else
      this.move(10);
  }
}
```

```
class main extends Object {
  void main() {
    Truck t = new Truck();
    Car c = new Car();
    Vehicle o = c;
    o.move(60);
    t.move(70);
    c.await(t);
  }
}
```



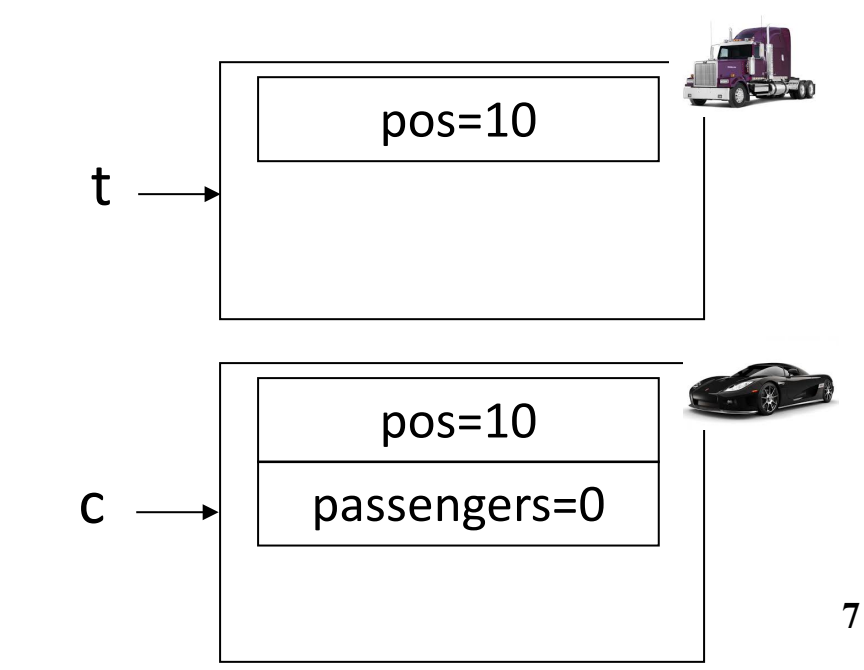
A Simple Example

```
class Vehicle extends object {
  int pos = 10;
  void move(int x) {
     pos = pos + x ;
  }
}

class Truck extends Vehicle {
  void move(int x){
    if (x < 55)
      pos = pos + x;
  }
}

class Car extends Vehicle {
  int passengers = 0;
  void await(vehicle v){
    if (v.pos < pos)
      v.move(pos - v.pos);
    else
      this.move(10);
  }
}
```

```
class main extends Object {
  void main() {
    Truck t = new Truck();
    Car c = new Car();
    Vehicle o = c;
    o.move(60);
    t.move(70);
    c.await(t);
  }
}
```



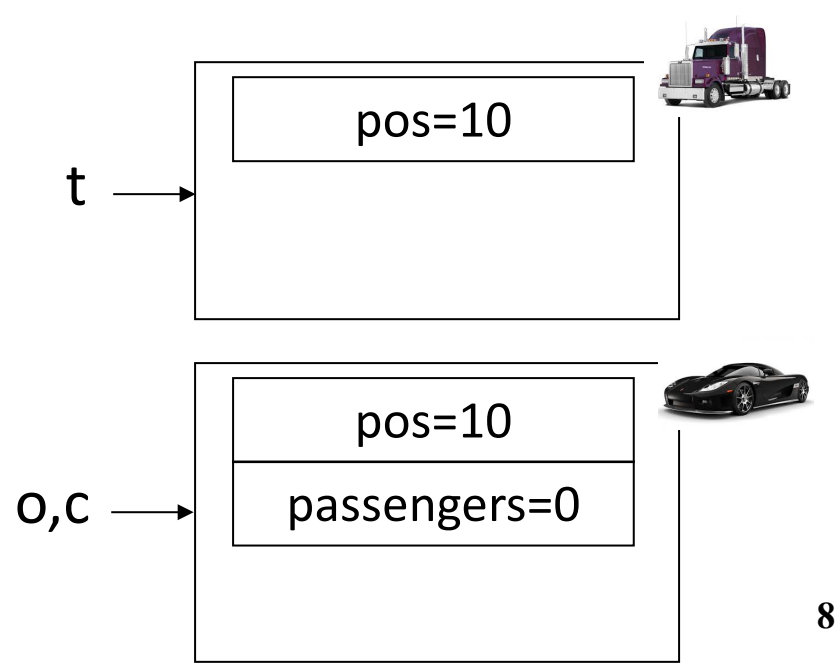
A Simple Example

```
class Vehicle extends object {
  int pos = 10;
  void move(int x) {
     pos = pos + x ;
  }
}

class Truck extends Vehicle {
  void move(int x){
    if (x < 55)
      pos = pos + x;
  }
}

class Car extends Vehicle {
  int passengers = 0;
  void await(vehicle v){
    if (v.pos < pos)
      v.move(pos - v.pos);
    else
      this.move(10);
  }
}
```

```
class main extends Object {
  void main() {
    Truck t = new Truck();
    Car c = new Car();
    Vehicle o = c;
    o.move(60);
    t.move(70);
    c.await(t);
  }
}
```



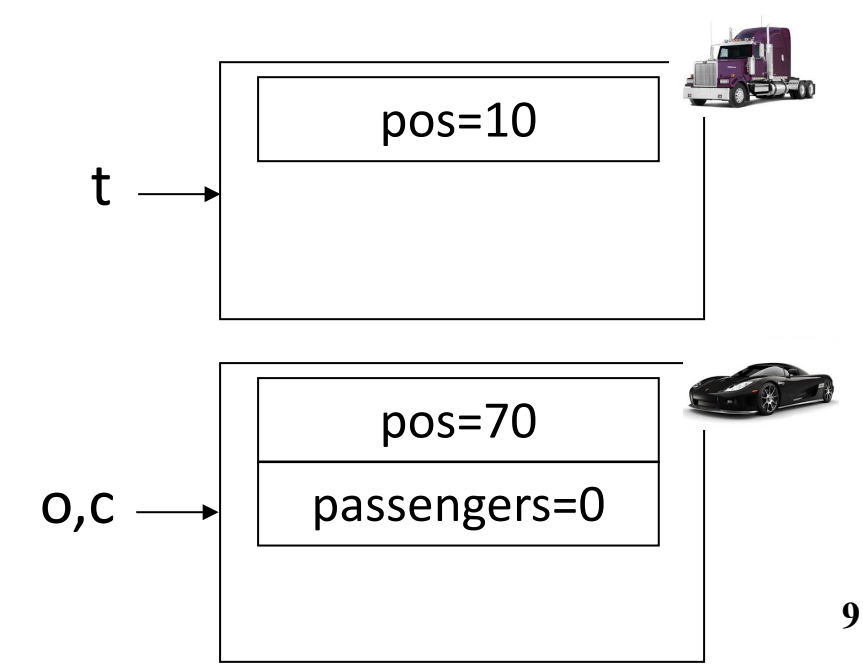
A Simple Example

```
class Vehicle extends object {
  int pos = 10;
  void move(int x) {
     pos = pos + x;
  }
}

class Truck extends Vehicle {
  void move(int x){
    if (x < 55)
      pos = pos + x;
  }
}

class Car extends Vehicle {
  int passengers = 0;
  void await(vehicle v){
    if (v.pos < pos)
      v.move(pos - v.pos);
    else
      this.move(10);
  }
}
```

```
class main extends Object {
  void main() {
    Truck t = new Truck();
    Car c = new Car();
    Vehicle o = c;
    o.move(60);
    t.move(70);
    c.await(t);
  }
}
```



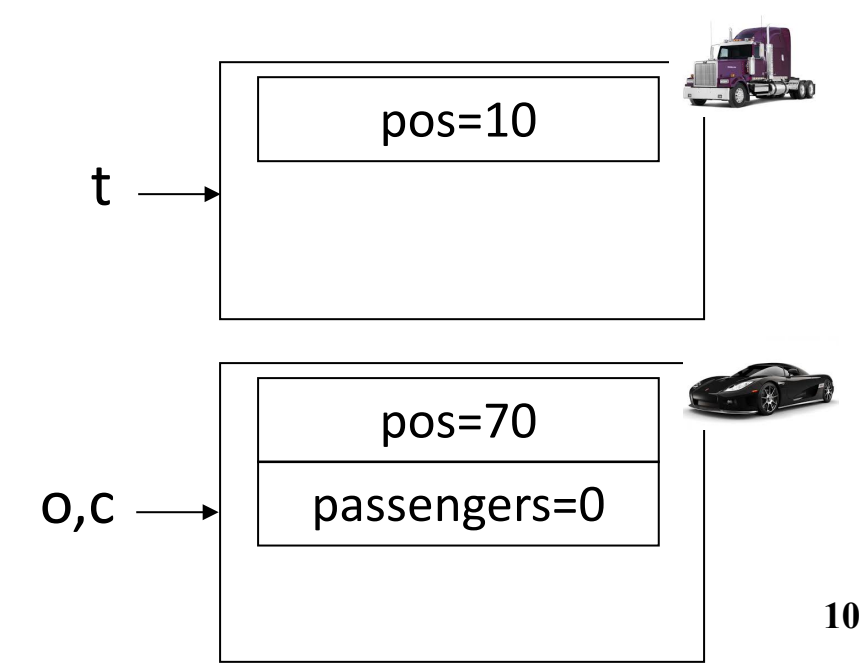
A Simple Example

```
class Vehicle extends object {
  int pos = 10;
  void move(int x) {
     pos = pos + x;
  }
}

class Truck extends Vehicle {
  void move(int x){
    if (x < 55)
      pos = pos + x;
  }
}

class Car extends Vehicle {
  int passengers = 0;
  void await(vehicle v){
    if (v.pos < pos)
      v.move(pos - v.pos);
    else
      this.move(10);
  }
}
```

```
class main extends Object {
  void main() {
    Truck t = new Truck();
    Car c = new Car();
    Vehicle o = c;
    o.move(60);
    t.move(70);
    c.await(t);
  }
}
```



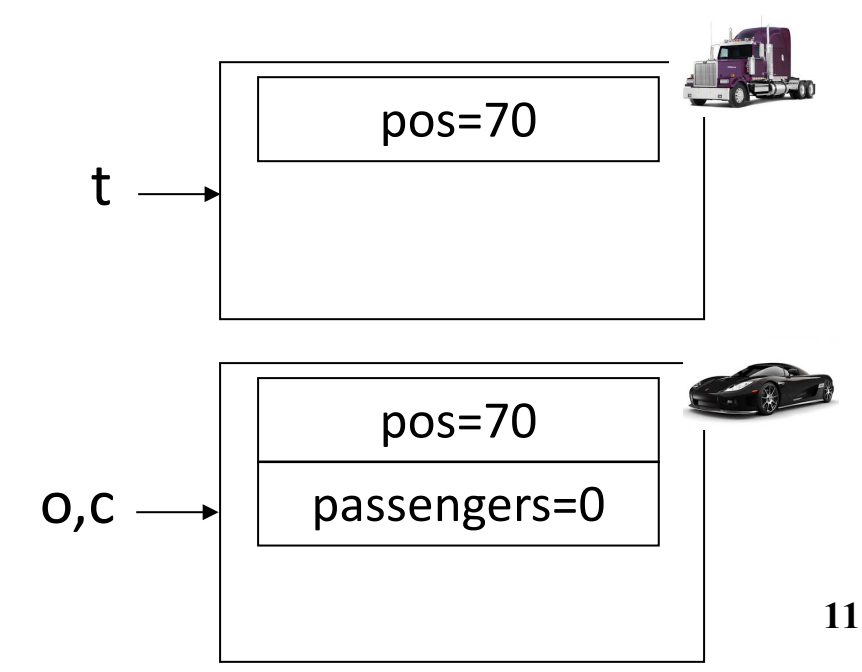
A Simple Example

```
class Vehicle extends object {
  int pos = 10;
  void move(int x) {
     pos = pos + x;
  }
}

class Truck extends Vehicle {
  void move(int x){
    if (x < 55)
      pos = pos + x;
  }
}

class Car extends Vehicle {
  int passengers = 0;
  void await(vehicle v){
    if (v.pos < pos)
      v.move(pos - v.pos);
    else
      this.move(10);
  }
}
```

```
class main extends Object {
  void main() {
    Truck t = new Truck();
    Car c = new Car();
    Vehicle o = c;
    o.move(60);
    t.move(70);
    c.await(t);
  }
}
```



Translation into C

Translation into C (Vehicle)

```
class Vehicle extends Object {
  int pos = 10;
  void move(int x) {
    pos = pos + x ;
  }
}
```

```
typedef struct Vehicle {
  int pos;
} Ve;
```

Translation into C (Vehicle)

```
class Vehicle extends Object {
  int pos = 10;
  void move(int x) {
    pos = pos + x ;
  }
}
```

```
typedef struct Vehicle {
  int pos;
} Ve;

void NewVe(Ve *this){
  this→pos = 10;
}

void moveVe(Ve *this, int x){
  this→pos = this→pos + x;
}
```

Translation into C (Truck)

```
class Truck extends Vehicle {
  void move(int x){
    if (x < 55)
      pos = pos + x;
  }
}
```

```
typedef struct Truck {
  int pos;
} Tr;

void NewTr(Tr *this){
  this→pos = 10;
}

void moveTr(Ve *this, int x){
  if (x<55)
    this→pos = this→pos + x;
}
```

Naïve Translation into C (Car)

```
class Car extends Vehicle {
 int passengers = 0;
 void await(vehicle v){
  if (v.pos < pos)
    v.move(pos – v.pos);
  else
    this.move(10);
 }
}
```

```
typedef struct Car{
  int pos;
  int passengers;
} Ca;

void NewCa (Ca *this){
  this→pos = 10;
  this→passengers = 0;
}

void awaitCa(Ca *this, Ve *v){
  if (v→pos < this→pos)
    moveVe(this→pos – v→pos)
  else
    MoveCa(this, 10)
}
```

Naïve Translation into C (Main)

```
class main extends object {
  void main() {
    Truck t = new Truck();
    Car c = new Car();
    Vehicle v = c;
    v.move(60);
    t.move(70);
    c.await(t);
  }
}
```

```
void mainMa(){
  Tr *t = malloc(sizeof(Tr));
  Ca *c = malloc(sizeof(Ca));
  Ve *v = (Ve*) c;
  moveVe(v, 60);
  moveVe(t, 70);
  awaitCa(c,(Ve*) t);
}
```

Naïve Translation into C (Main)

```
class main extends object {
  void main() {
    Truck t = new Truck();
    Car c = new Car();
    Vehicle v = c;
    v.move(60);
    t.move(70);
    c.await(t);
  }
}
```

```
void mainMa(){
  Tr *t = malloc(sizeof(Tr));
  Ca *c = malloc(sizeof(Ca));
  Ve *v = (Ve*) c;
  moveVe(v, 60);
  moveVe(t, 70);
  awaitCa(c,(Ve*) t);
}
```

```
void moveCa() ?
```

Naïve Translation into C (Main)

```
class main extends object {
  void main() {
    Truck t = new Truck();
    Car c = new Car();
    Vehicle v = c;
    v.move(60);
    t.move(70);
    c.await(t);
  }
}
```

```
void mainMa(){
  Tr *t = malloc(sizeof(Tr));
  Ca *c = malloc(sizeof(Ca));
  Ve *v = (Ve*) c;
  moveVe(v, 60);
  moveVe(t, 70);
  awaitCa(c,(Ve*) t);
}
```

```
void moveCa() ?
```

```
void moveVe(Ve *this, int x){
  this→pos = this→pos + x;
}
```

Naïve Translation into C (Main)

```
class main extends object {
  void main() {
    Truck t = new Truck();
    Car c = new Car();
    Vehicle v = c;
    v.move(60);
    t.move(70);
    c.await(t);
  }
}
```

```
typedef struct Vehicle {
  int pos;
} Ve;
```

```
typedef struct Car{
  int pos;
  int passengers;
} Ca;
```

```
void mainMa(){
  Tr *t = malloc(sizeof(Tr));
  Ca *c = malloc(sizeof(Ca));
  Ve *v = (Ve*) c;
  moveVe(v, 60);
  moveVe(t, 70);
  awaitCa(c,(Ve*) t);
}
```

```
void moveCa() ?
```

```
void moveVe(Ve *this, int x){
  this→pos = this→pos + x;
}
```

Naïve Translation into C (Main)

```
class main extends object {
  void main() {
    Truck t = new Truck();
    Car c = new Car();
    Vehicle v = c;
    v.move(60);
    t.move(70);
    c.await(t);
  }
}
```

Vehicle x = t;
x.move(20);

```
void mainMa(){
  Tr *t = malloc(sizeof(Tr));
  Ca *c = malloc(sizeof(Ca));
  Ve *v = (Ve*) c;
  moveVe(v, 60);
  moveVe(t, 70);
  awaitCa(c,(Ve*) t);
}
```

```
Ve *x = t;
moveTr((Tr*)x, 20);
```

Naïve Translation into C (Main)

```
class main extends object {
  void main() {
    Truck t = new Truck();
    Car c = new Car();
    Vehicle v = c;
    v.move(60);
    t.move(70);
    c.await(t);
  }
}
```

Vehicle x = t;
x.move(20);

```
void mainMa(){
  Tr *t = malloc(sizeof(Tr));
  Ca *c = malloc(sizeof(Ca));
  Ve *v = (Ve*) c;
  moveVe(v, 60);
  moveVe(t, 70);
  awaitCa(c,(Ve*) t);
}
```

```
Ve *x = t;
moveTr((Tr*)x, 20);
```

```
void moveVe(Ve *this, int x){…}
```

```
void moveTr(Ve *this, int x){…}
```

Translation into C

Compiling Simple Classes

- Fields are handled as records
- Methods have unique names

```
class A {
   field a1;
   field a2;
   method m1() {…}
   method m2(int i) {…}
}
```

Runtime object

a1
a2

Compile-Time Table

m1A
m2A

```
void m2A(classA  *this, int i) {
   // Body of m2 with any object
   // field f as this->f
   …
}
```

Compiling Simple Classes

- Fields are handled as records
- Methods have unique names

```
class A {
   field a1;
   field a2;
   method m1() {…}
   method m2(int i) {…}
}
```

Runtime object

a1
a2

Compile-Time Table

m1A
m2A

```
a.m2(5)
```

```
m2A(a,5)
```

```
void m2A(classA  *this, int i) {
   // Body of m2 with any
   // object-field f as this f
   …
}
```

Features of OO languages

- **Inheritance**
  - Subclass gets (inherits) properties of superclass
- **Method overriding**
  - Multiple methods with the same name with different signatures
- **Abstract** (aka **virtual**) **methods**
- **Polymorphism**
  - Multiple methods with the same name and different signatures but with different implementations
- **Dynamic dispatch**
  - Lookup methods by (runtime) type of target object

Compiling OO languages

- “Translation into C”
- Powerful runtime environment

- Adding “gluing” code