The "this" reference

```
public class Car {
    double speed;
    Car()
    {
        speed = 0.0;
    }
    void accelerate(double amount)
    {
        speed = speed + amount;
    }
}

public class MyCarSimulation {
  public static void main(String[] args)
  {
    Car mycar = new Car();
    Car yourcar = new Car();
    mycar.accelerate(90.0);
    yourcar.accelerate(100.0);
  }
}
```

The "this" reference

```
public class Car {
    double speed;
    Car()
    {
        this.speed = 0.0;
    }
    void accelerate(double amount)
    {
        this.speed = this.speed + amount;
    }
}

public class MyCarSimulation {
  public static void main(String[] args)
  {
    Car mycar = new Car();
    Car yourcar = new Car();
    mycar.accelerate(90.0);
    yourcar.accelerate(100.0);
  }
}
```

The "this" reference and objects as arguments

```
public class Car
{
    double speed;

    Car()
    {
        this.speed = 0.0;
    }
    void accelerate(double amount)
    {
        this.speed = this.speed + amount;
    }
    boolean faster_than(Car other_car)
    {
        return this.speed > other_car.speed;
    }
}
```

The “this” reference and objects as arguments

```
public class MyCarSimulation {
  public static void main(String[] args)
  {
    Car mycar = new Car();
    Car yourcar = new Car();
    mycar.accelerate(90.0);
    yourcar.accelerate(100.0);

    if (mycar.faster_than(yourcar)) {
      System.out.println(“I\’m faster than you”);
    }
    else {
      System.out.println(“You are faster than me”)
    }
  }
}
```

The "this" reference and objects as arguments

```
public class Car
{
    double speed;

    Car()
    {
        speed = 0.0;
    }
    void accelerate(double amount)
    {
        speed = speed + amount;
    }
    boolean faster_than(Car other_car)
    {
        return speed > other_car.speed;
    }
}
```

The "this" reference

```
public class K
{
    int n;

    void p(int n)
    {
        this.n = n;  // Use this. to distinguish
    }                // the n's
    void q()
    {
        int n = 5;
        this.n = n;
        // ...
    }
    void r()
    {
        n = n * 2; // Same as this.n=this.n*2;
    }
}
```

Aristotle

- Silogisms:
  - **If** every city *has a* mayor, **and** Edinburgh *is a* city, **then** Edinburgh *has a* mayor.
  - **If** every car *has an* engine, **and** this *is a* car, **then** this *has an* engine.
  - **If** every A *has a* B, **and** x *is an* A, **then** x *has a* B.

- In OOP:
  - **If** every object of type A *has an* attribute of type B **and** x *is an* A object **then** x *has an* attribute of type B.
  - **If** a class A *has an* attribute of class B, **and** x *is an* instance of A, **then** x *has an* attribute of class B.

Objects and Aggregation

- Objects are data with structure: objects have attributes.
- We think of attributes as characteristics of objects in a class.
- The relation between an object and its attributes can be seen as a "has a" relationship.
- Aggregation is the composition of objects in different parts or *aggregates* (the attributes.)

Objects and Aggregation

- Aggregation is given by the "has a" relationship.

```
public class A {
    B u;
    // ...
}
public class C {
    void m()
    {
        A x = new A();
        ... x.u ...
    }
}
```

Objects and Aggregation

```
public class Mayor {
    // ...
}
public class City {
    Mayor mayor;
    // ...
}
public class Something {
    void p()
    {
        City edinburgh = new City();
        edinburgh.mayor = new Mayor();
    }
}
```

Objects and Aggregation

```
public class Engine {
    // ...
}
public class Car {
    Engine engine;
    // ...
}
public class Something {
    void p()
    {
        Car mycar = new Car();
        mycar.engine = new Engine();
    }
}
```

Example

```
public class Engine
{
    private boolean on;
    private double rpm;

    public Engine()
    {
        on = false;
        rpm = 0.0;
    }
    public void turn_on()
    {
        on = true;
        rpm = 50.0;
    }
    public void accelerate()
    {
        rpm = rpm + 10.0;
    }
```

```
    public void decelerate()
    {
        rpm = rpm - 10.0;
    }
    public double get_rpm()
    {
        return rpm;
    }
}
```

Example (contd.)

```
public class Car
{
    private Engine engine;
    private double speed;

    public Car()
    {
        engine = new Engine();
        speed = 0.0;
    }
    public void turn_on()
    {
        engine.turn_on();
    }
    public void acelerate()
    {
        engine.acelerate();
        speed = speed + 10 * engine.get_rpm();
    }
}
```

Mutual reference

```
public class BankAccount
{
    private float balance;
    private Person owner;

    public BankAccount(Person owner)
    {
        this.owner = owner;
        balance = 0.0;
    }
    public void deposit(float amount)
    {
        balance = balance + amount;
    }
    public void withdraw(float amount)
    {
        if (amount <= balance)
            balance = balance - amount;
    }
    public float balance() { return balance; }
    public Person owner()  { return owner; }
}
```

Mutual reference

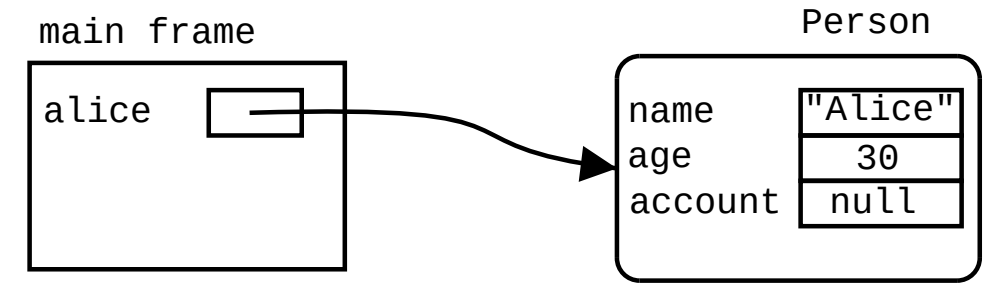
```
public class Person
{
    private String name;
    private int age;
    private BankAccount account;

    public Person(String name, int age)
    {
        this.name = name;
        this.age = age;
        account = null;
    }
    public void open_account(BankAccount a)
    {
        account = a;
    }
    public void open_account()
    {
        account = new BankAccount(this);
    }
    // Continues below...
```

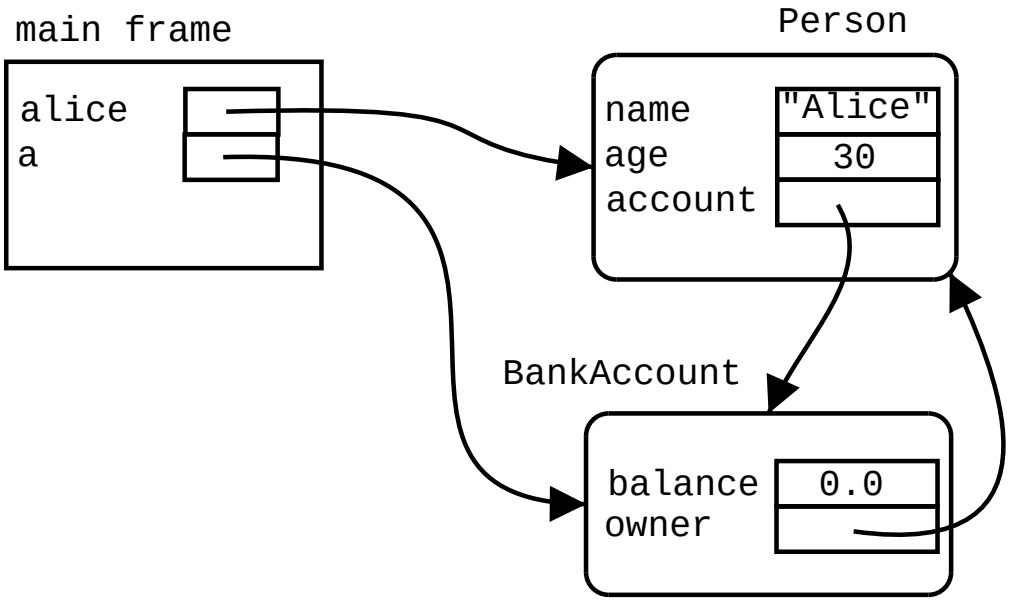
```
    public String name()
    {
        return name;
    }
    public BankAccount account()
    {
        return account;
    }
}
```

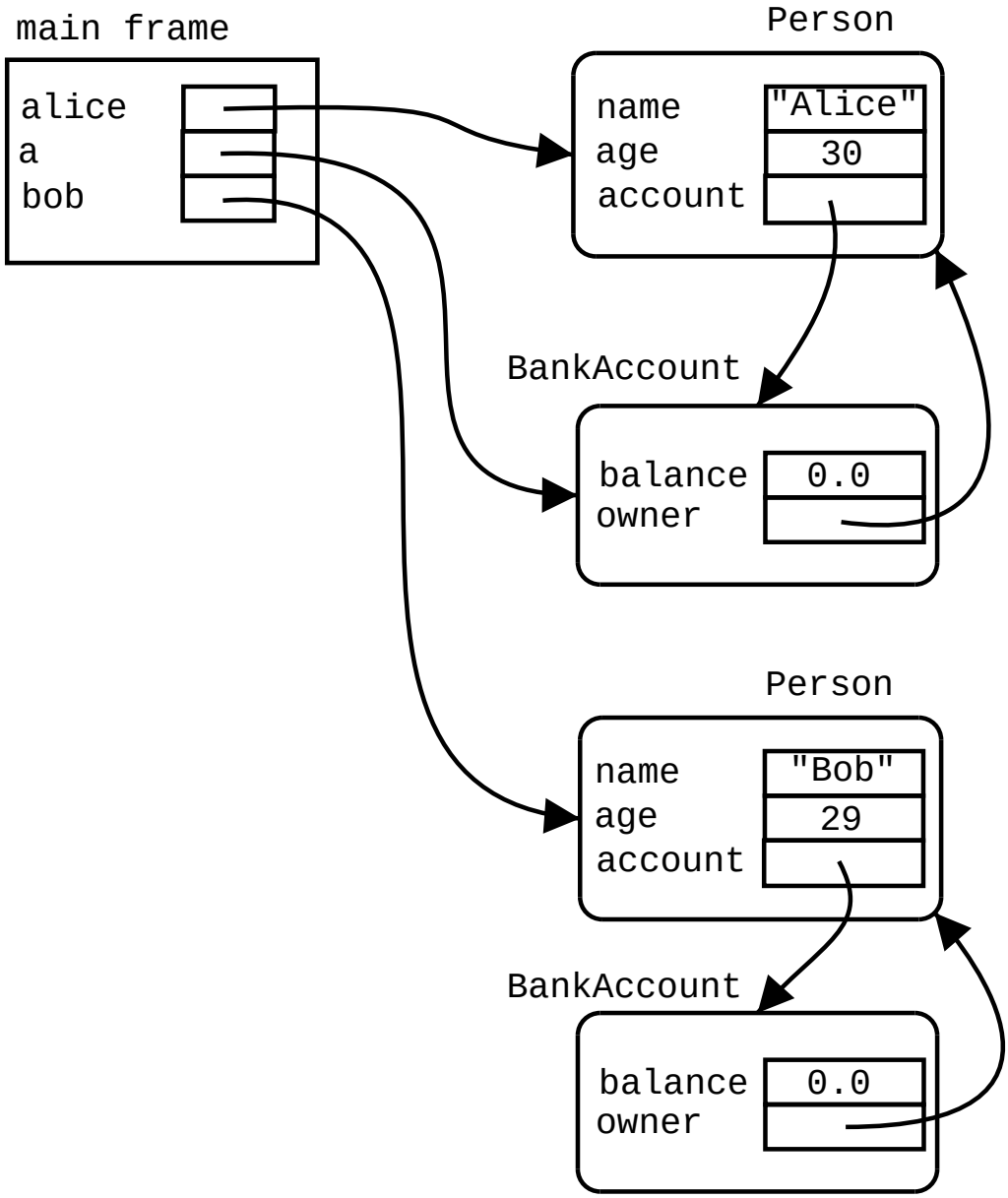
Mutual reference (contd.)

```
public class Banking
{
  public static void main(String[] args)
  {
    Person alice = new Person(“Alice”, 30);
    BankAccount a = new BankAccount(alice);
    alice.open_account(a);

    Person bob = new Person(“Bob”, 29);
    bob.open_account();

    BankAccount b = bob.account();
    b.deposit(300.0);

    alice.account().deposit(200.0f);

    System.out.println(b.balance());
    System.out.println(alice.account().balance());
    System.out.println(a.balance());
  }
}
```

Mutual reference






main frame
alice
a
bob
Person
name
age
account
"Alice"
30
BankAccount
balance
owner
0.0
Person
name
age
account
"Bob"
29
BankAccount
balance
owner
0.0

Mutual reference

```
public class Person
{
    private String name;
    private int age;
    private Person spouse;

    public Person(String name, int age)
    {
        this.name = name;
        this.age  = age;
        this.spouse = null;
    }
    public void marry(Person someone)
    {
        this.spouse = someone;
        someone.spouse = this;
    }
    public String name()   { return name; }
    public Person spouse() { return spouse; }
}
```

Mutual reference (contd.)

```
public class Marriage
{
    public static void main(String[] args)
    {
        Person a = new Person(“Alice”, 30);
        Person b = new Person(“Bob”, 29);
        a.marry(b);
        System.out.println(a.name());
        System.out.println(a.spouse().name());
        System.out.println(b.name());
        System.out.println(b.spouse().name());
    }
}
```