Object-Oriented Programming

Week 2, Spring 2013

Classes

Weng Kai

What is an object?

What is an object?

- Object = Entity

What is an object?

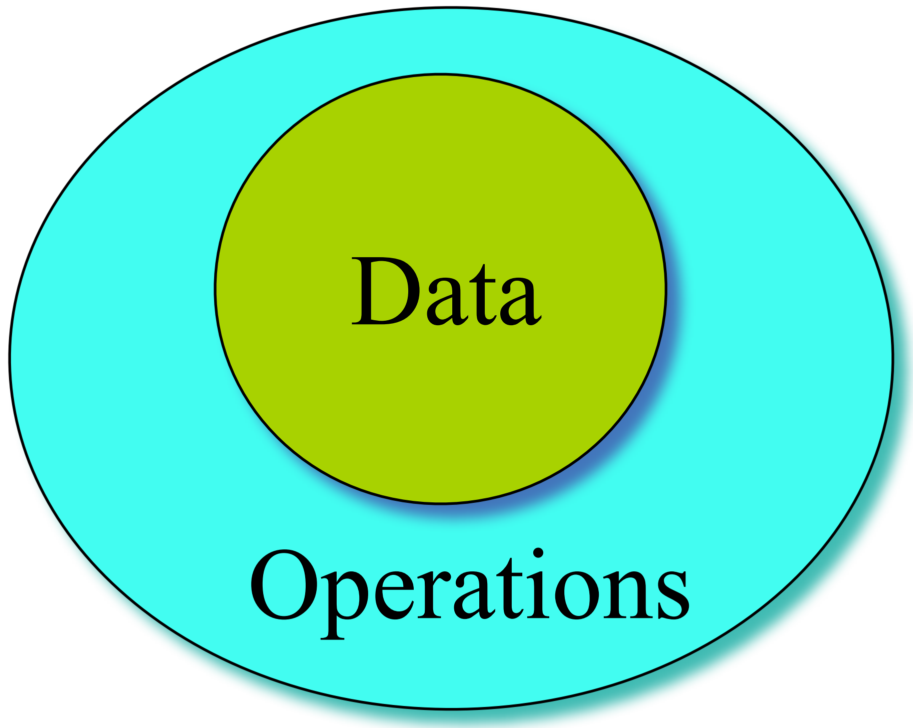
- Object = Entity
- Object may be
  - Visible or
  - invisible

What is an object?

- Object = Entity
- Object may be
  - Visible or
  - invisible
- Object is variable in programming languages.

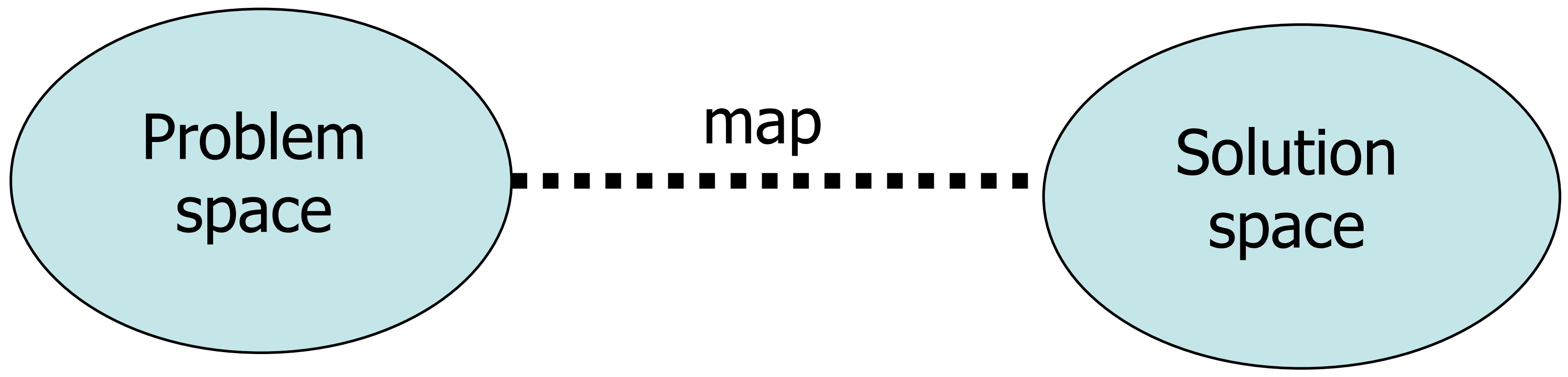
Objects = Attributes + Services

- Data: the properties or status
- Operations: the functions



Mapping

- From the problem space to the solution one.



Procedural Languages

- C doesn’t support relationship btw data and functions.

```
typedef struct point3d {
    float x;
    float y;
    float z;
} Point3d;

void Point3d_print(const Point3d* pd);
Point3d a;
a.x = 1; a.y = 2; a.z=3;
Point3d_print(&a);
```

C++ version

```
class Point3d {
public:
    Point3d(float x,float y,float z);
    print();
private:
    float x;
    float y;
    float z;
} ;


Point3d a(1,2,3);
a.print();
```

C vs. C++

```
typedef struct point3d {
    float x;
    float y;
    float z;
} Point3d;

void Point3d_print(const
Point3d* pd);

Point3d a;
a.x = 1; a.y = 2; a.z=3;
Point3d_print(&a);
```

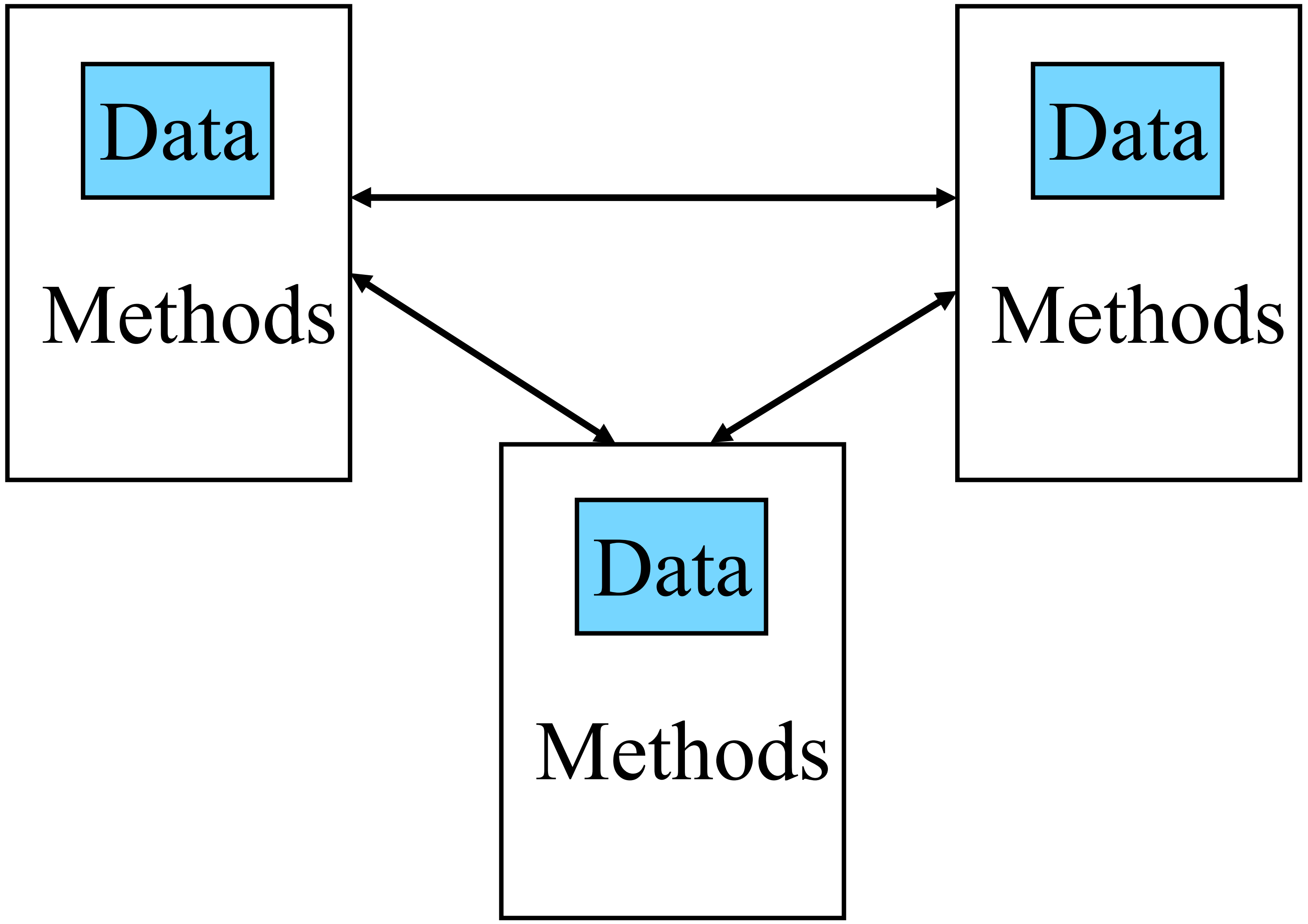
```
class Point3d {
public:
    Point3d(float
x,float y,float z);
    print();
private:
    float x;
    float y;
    float z;
} ;

Point3d a(1,2,3);
a.print();
```

What is object-oriented

- A way to organize
  - Designs
  - Implementations
- Objects, not control or data flow, are the primary focus of the design and implementation.
- To focus on things, not operations.

Object Oriented Programming

- Objects send and receive messages (objects do things!)

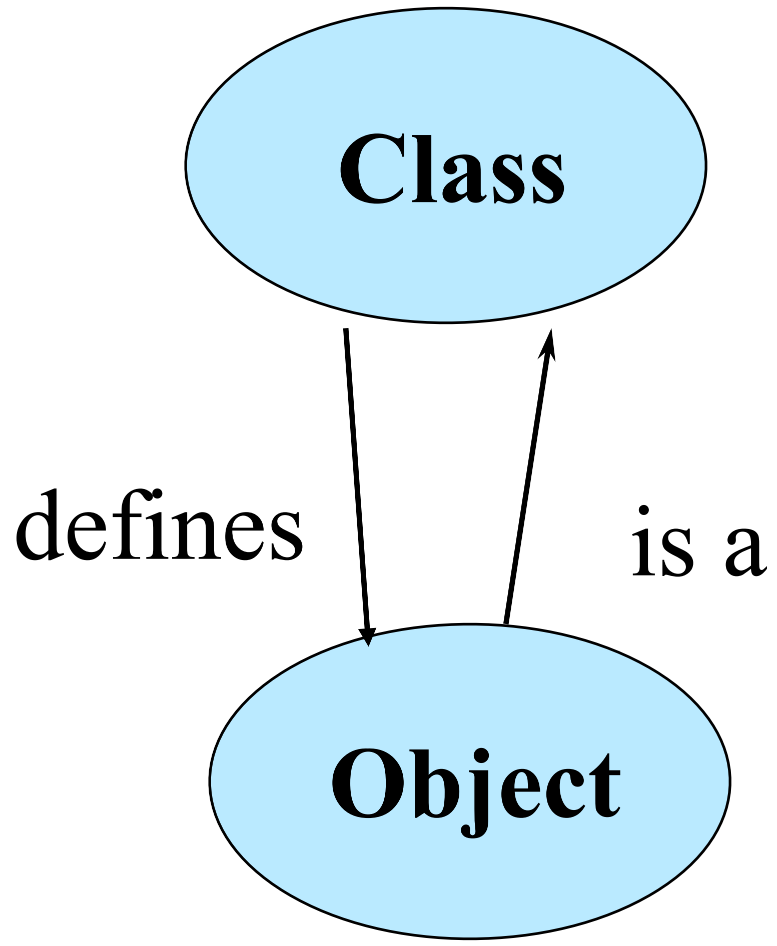


Objects send messages

- Messages are
  - *Composed* by the sender
  - *Interpreted* by the receiver
  - *Implemented* by methods
- Messages
  - May cause receiver to change state
  - May return results

Object vs. Class

- Objects (cat)
  - Represent things, events, or concepts
  - Respond to messages at run-time
- Classes (cat class)
  - Define properties of instances
  - Act like types in C++



OOP Characteristics

1. Everything is an object.
2. A program is a bunch of objects telling each other what to do by sending messages.
3. Each object has its own memory made up of other objects.
4. Every object has a type.
5. All objects of a particular type can receive the same messages.

An object has an interface

- The interface is the way it receives messages.
- It is defined in the class the object belong to.

合格证

雅牌 节能灯
7W220V50Hz

Functions of the interface

- Communication
- Protection

The Hidden Implementation

- Inner part of an object, data members to present its state, and the actions it takes when messages is rcvd is hidden.
- Class creators vs. Client programmers
  - Keep client programmers' hands off portions they should not touch.
  - Allow the class creators to change the internal working of the class without worrying about how it will affect the client programmers.

Encapsulation

- bundle data and methods dealing with these data together in an object
- Hide the details of the data and the action
- Restrict only access to the publicized methods.

Ticket Machine

- Ticket machines print a ticket when a customer inserts the correct money for their fare.
- Our ticket machines work by customers 'inserting' money into them, and then requesting a ticket to be printed. A machine keeps a running total of the amount of money it has collected throughout its operation.

Wo möchten Sie hin? Berlin und Brandenburg gibt's hier!
Where would you like to go? Berlin and Brandenburg is available here!
Fahrscheine
Tickets
BVG

Procedure-Oriented

- Step to the machine
- Insert money into the machine
- The machine prints a ticket
- Take the ticket and leave

Wo möchten Sie hin? Berlin und Brandenburg gibt's hier!
Where would you like to go? Berlin and Brandenburg is available here!
Fahrscheine
Tickets
BVG

Procedure-Oriented

- Step to the machine
- Insert money into the machine
- The machine prints a ticket
- Take the ticket and leave

We make a program simulates the procedure of buying tickets. It works. But there is no such machine. There's nothing left for the further development.

Wo möchten Sie hin? Berlin und Brandenburg gibt's hier!
Where would you like to go? Berlin and Brandenburg is available here!
Fahrscheine
Tickets
BVG

Procedure-Oriented

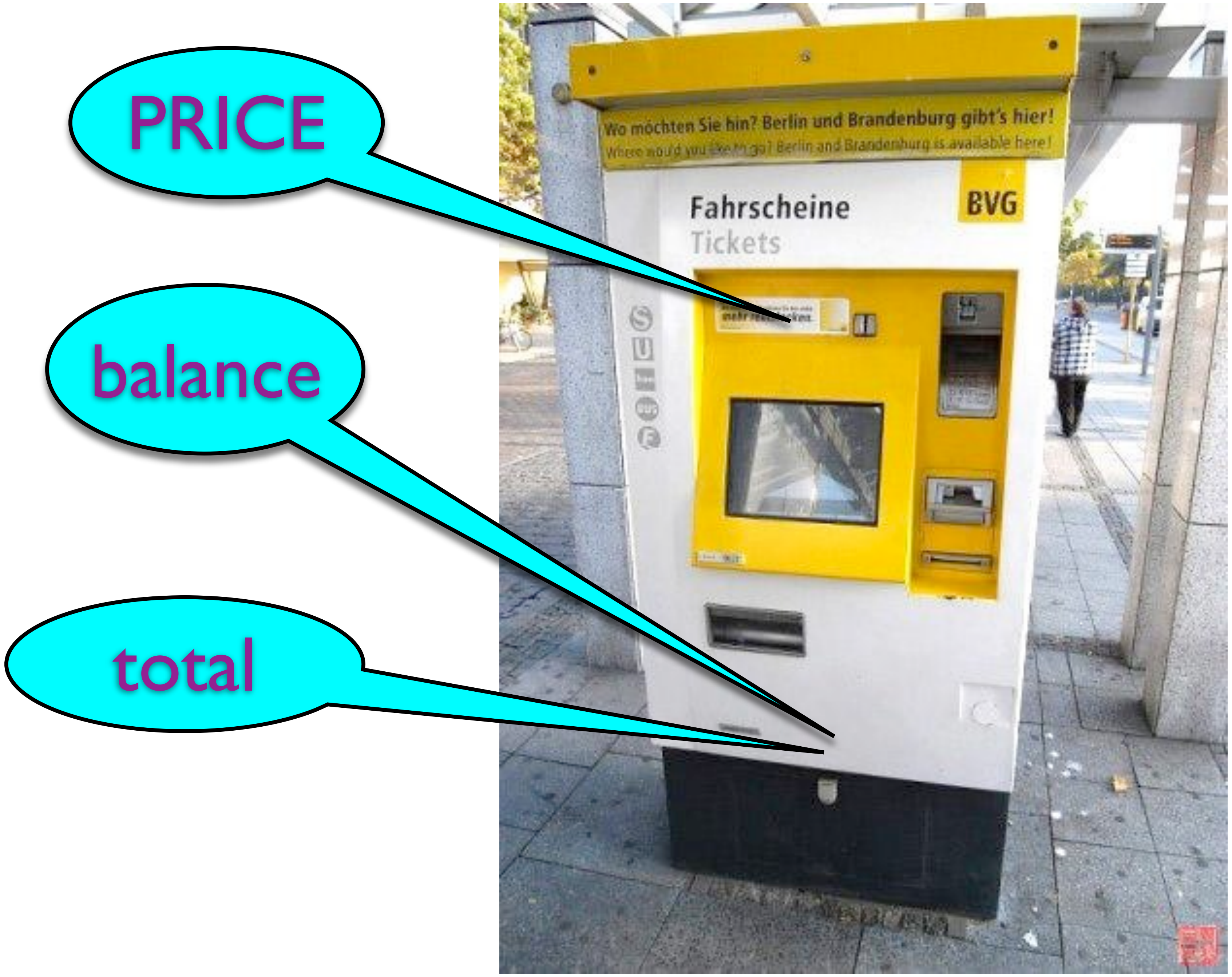
- Step to the machine
- Insert money into the machine
- The machine prints a ticket
- Take the ticket and leave

We make a program simulates the procedure of buying tickets. It works. But there is no such machine. There's nothing left for the further development.

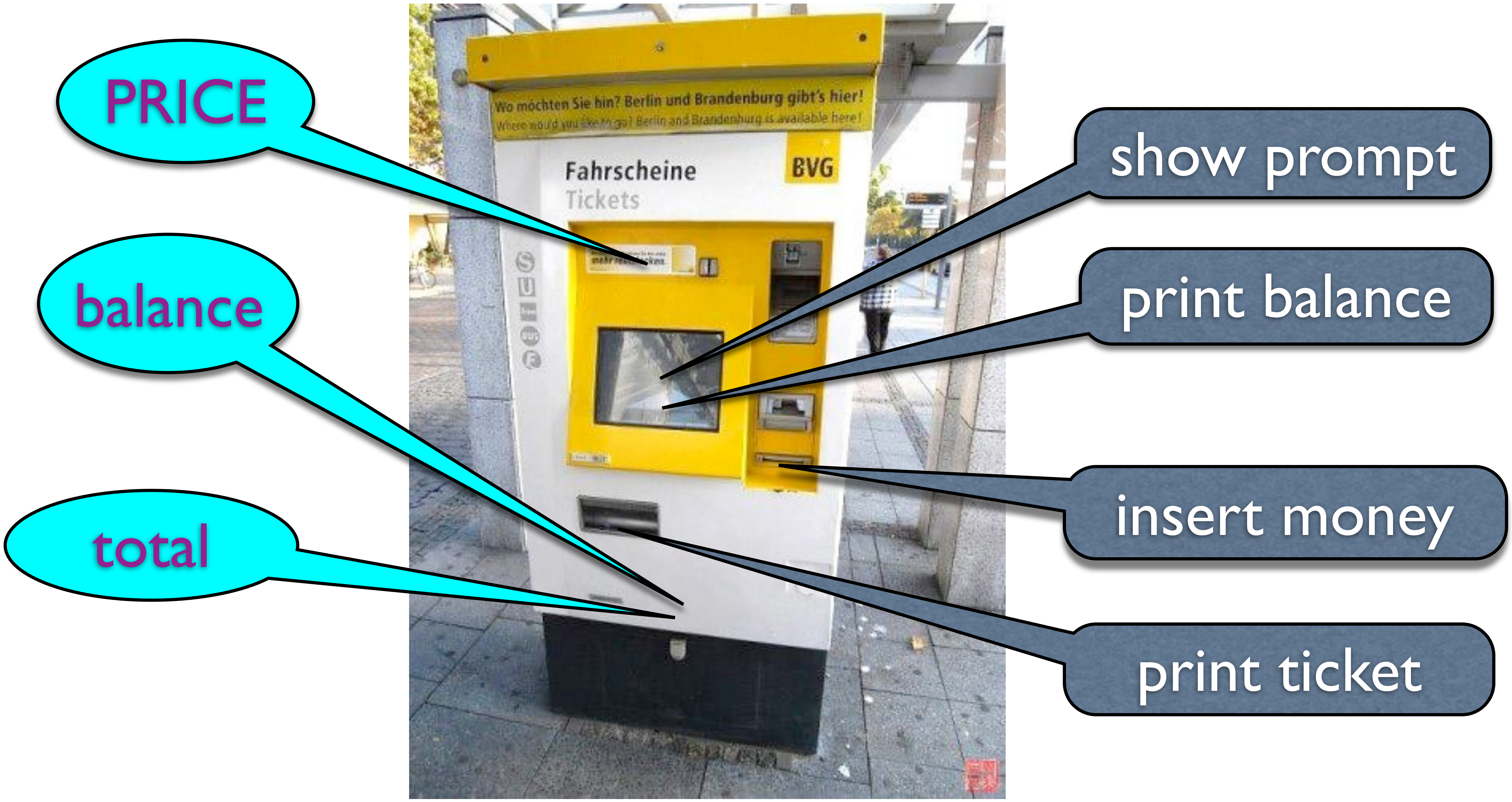
Something is there

Wo möchten Sie hin? Berlin und Brandenburg gibt's hier!
Where would you like to go? Berlin and Brandenburg is available here!
Fahrscheine
Tickets
BVG

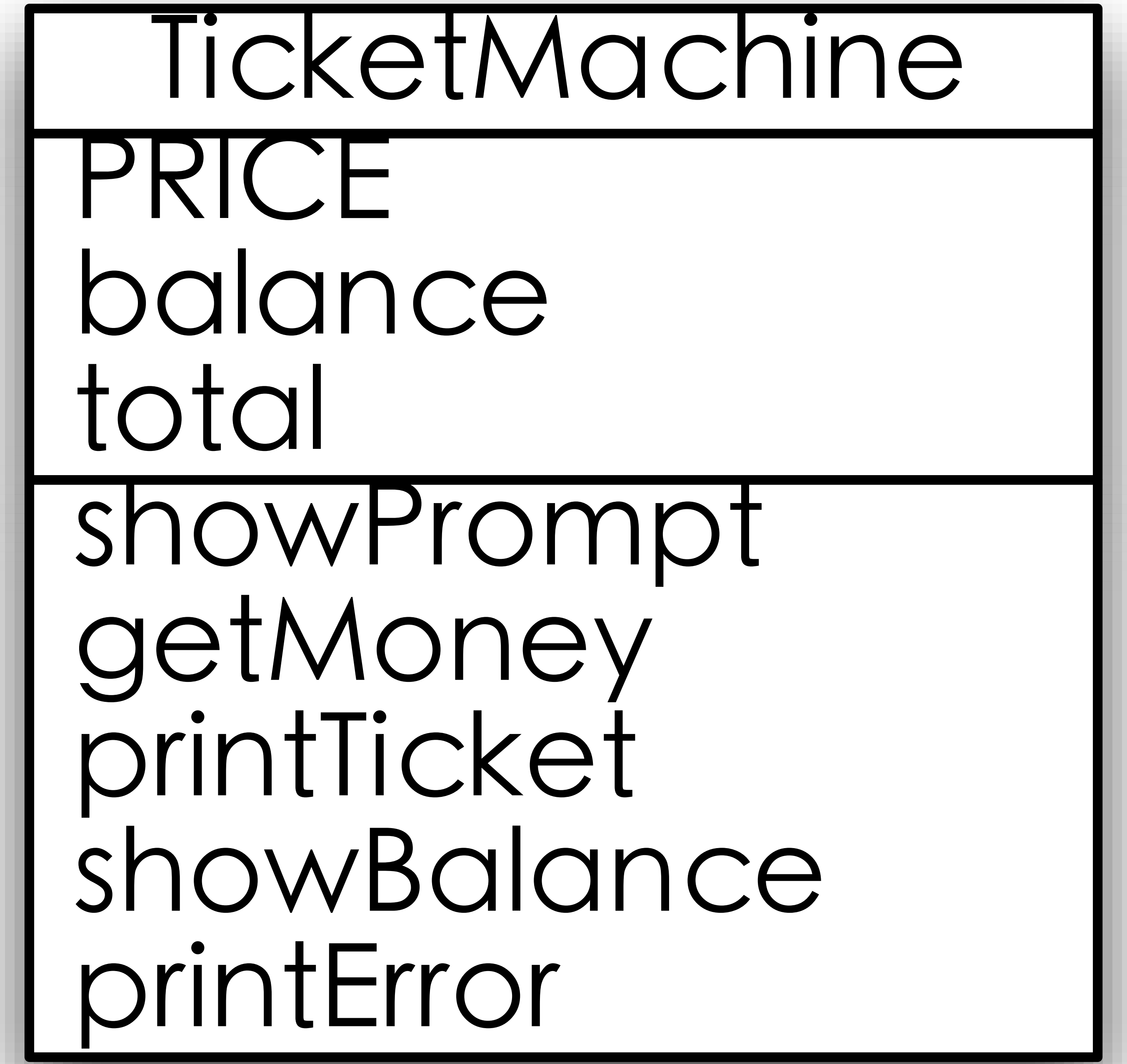
Something is there


PRICE
balance
total
Wo möchten Sie hin? Berlin und Brandenburg gibt's hier!
Where would you like to go? Berlin and Brandenburg is available here!
Fahrscheine
Tickets
BVG

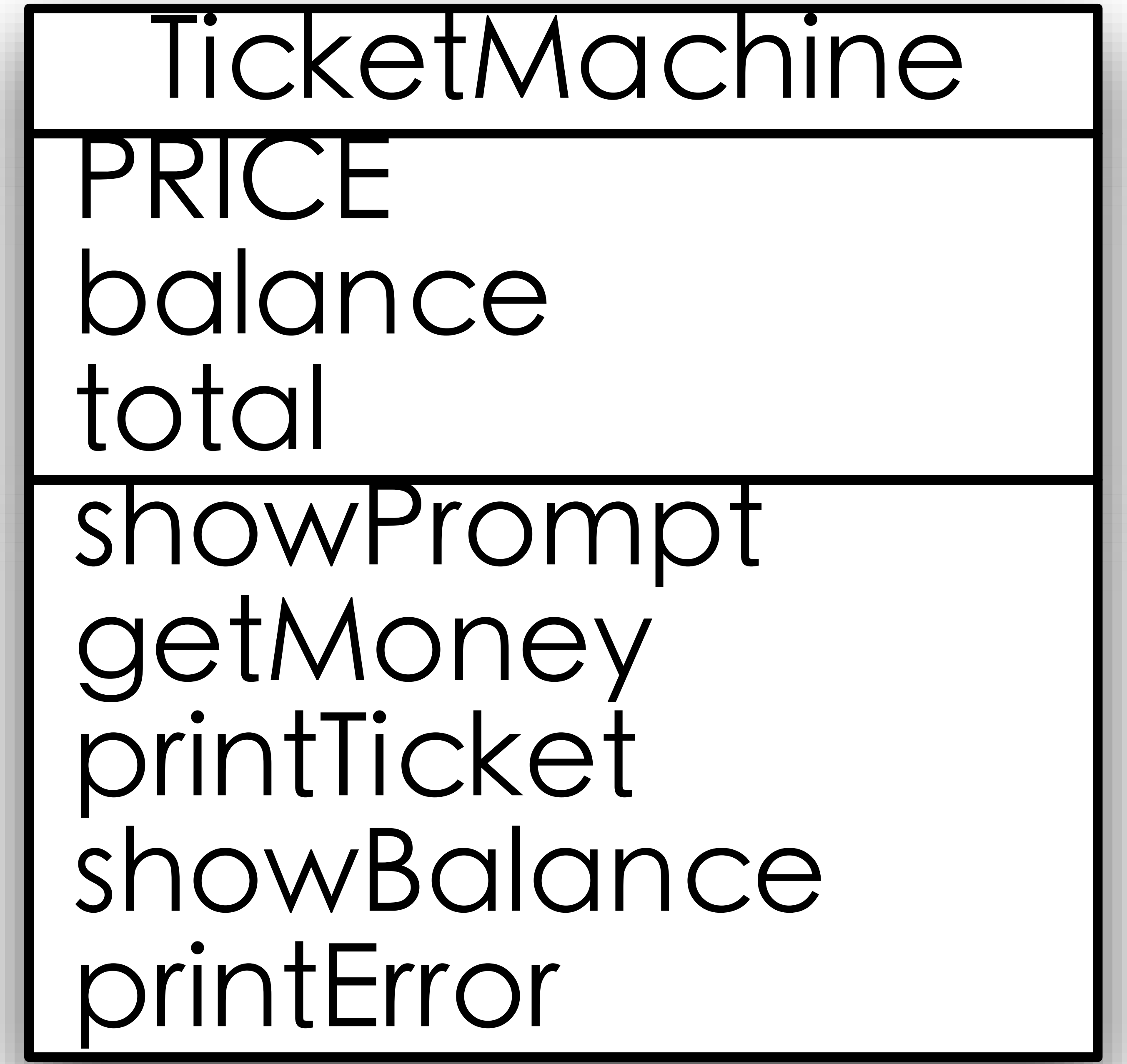
Something is there

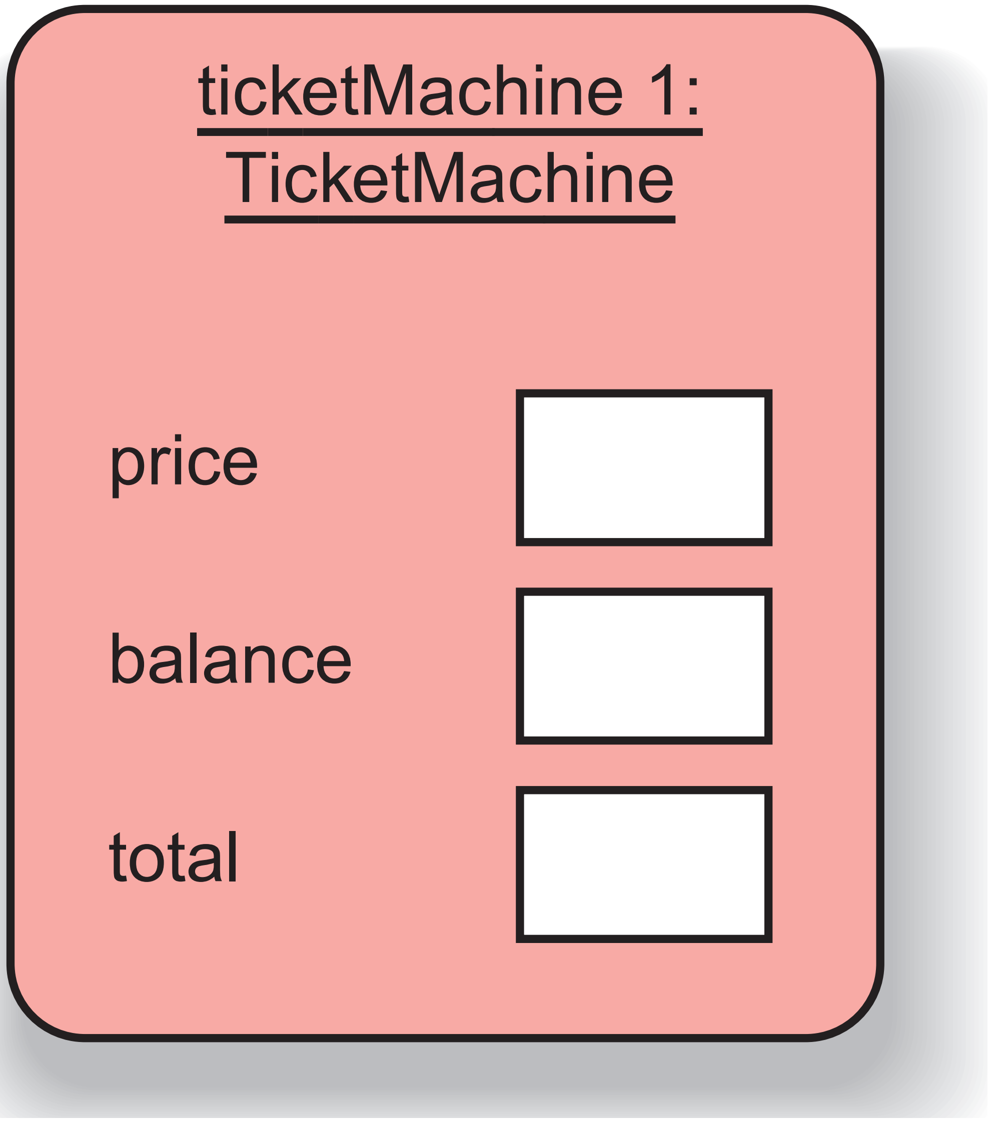


Something is here

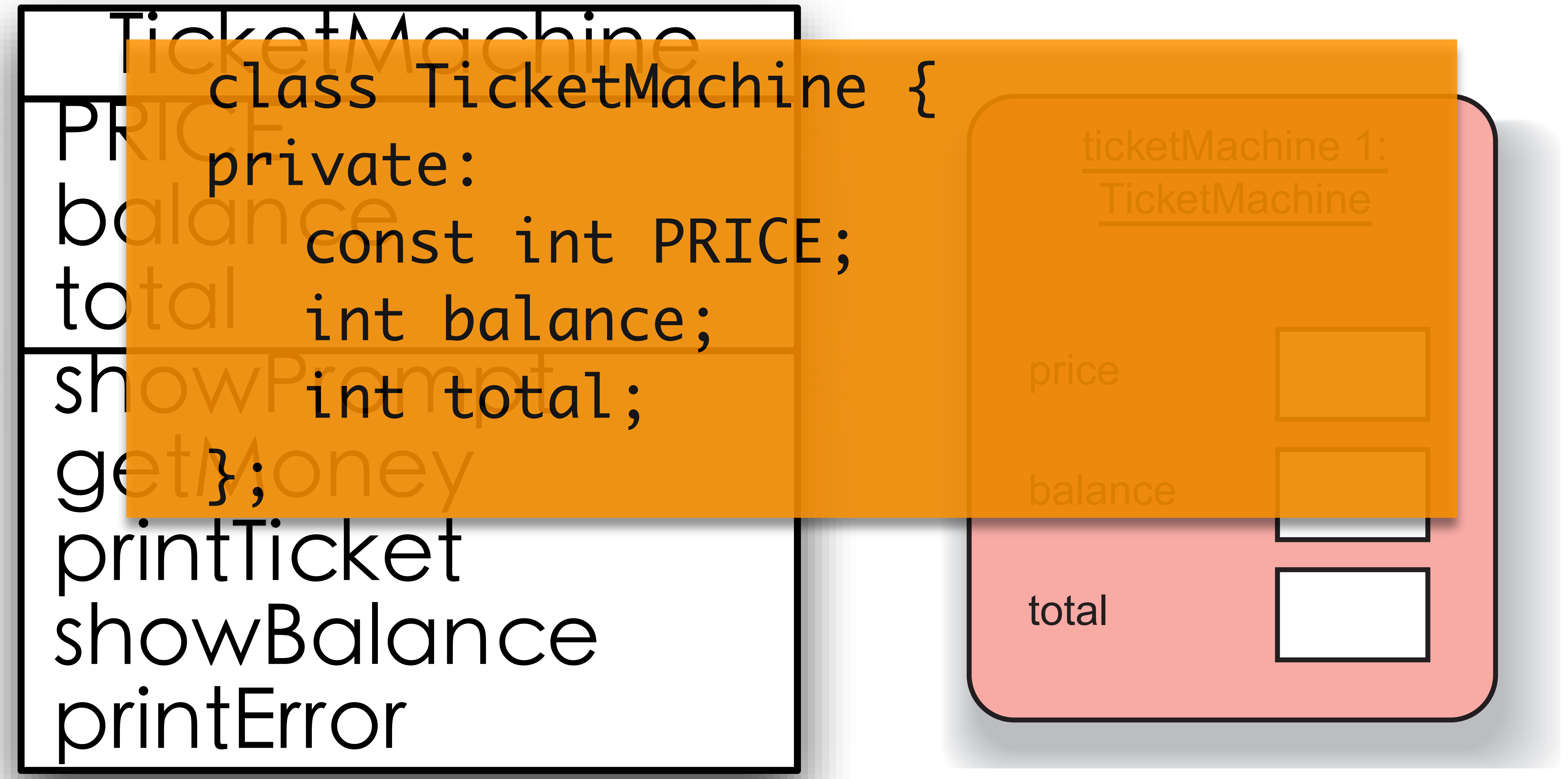


Something is here



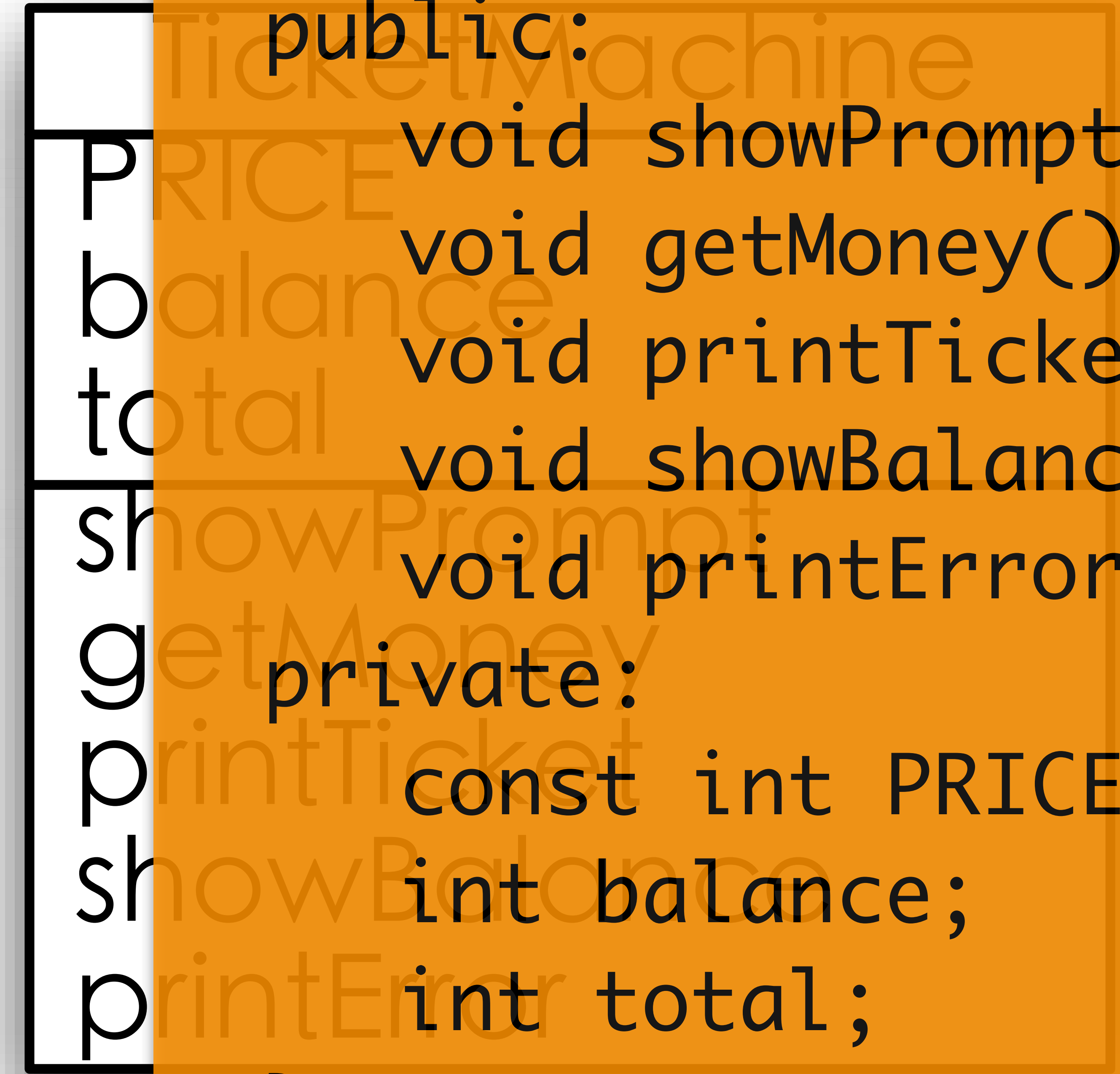


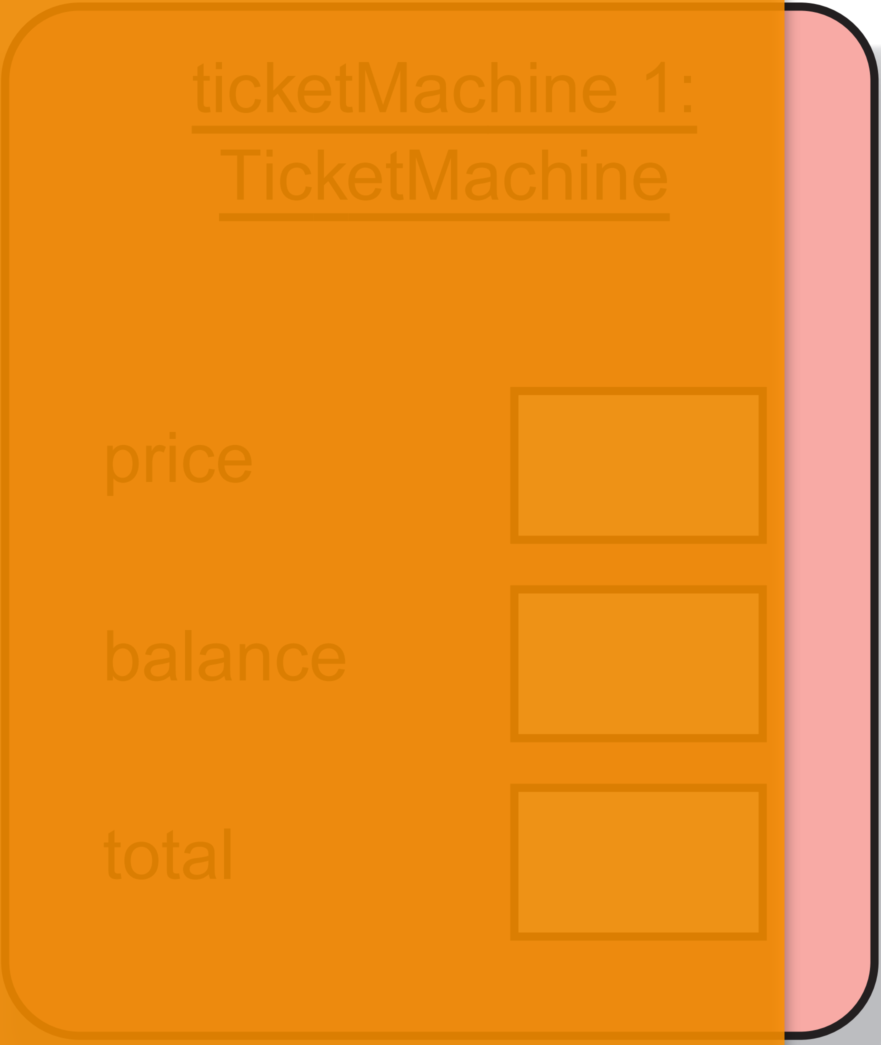
Turn it into code



```
class TicketMachine {
private:
    const int PRICE;
    int balance;
    int total;
};
```

Turn it into code





```
class TicketMachine {
public:
    void showPrompt();
    void getMoney();
    void printTicket();
    void showBalance();
    void printError();
private:
    const int PRICE;
    int balance;
    int total;
};
```