CS 101 Computer Programming and Utilization

Lecture 13

Classes

Mar 1, 2011

Prof. R K Joshi
Computer Science and Engineering
IIT Bombay
Email: rkj@cse.iitb.ac.in

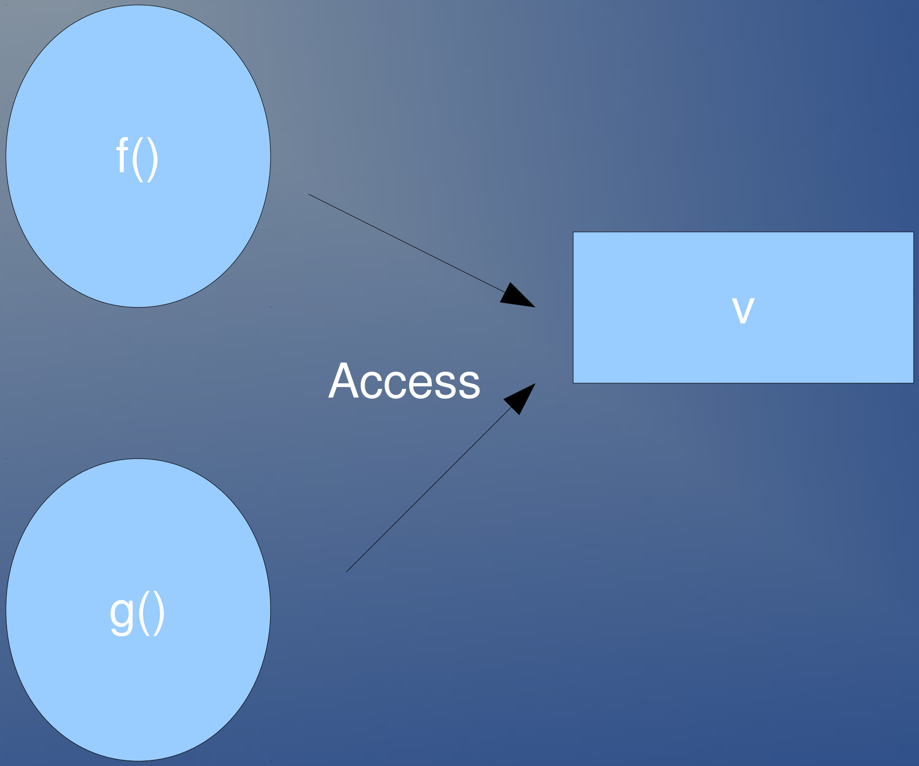
Revision

- keywords, identifiers
- variables
- types
- expressions, statements, main
- assignment, equality, comparison, logical operations
- storage allocation
- arrays, index
- dynamic allocation
- functions
- pointers for arrays
- syntax and grammar
- sequential flow
- branching
- iteration and iterative refinement
- recursion
- input output
- separate compilation
- parameter passing
- errors and debugging
- making flowcharts
- lots of examples and actual programming labs with practice

A case of many functions sharing a variable

- We know that a function can use a global variable
  - e.g. in the case of counting no. of calls to Fibonacci
  - sharing between many invocations of one function
- More, we can extend this ability to sharing among many invocations of many functions

sharing between many functions



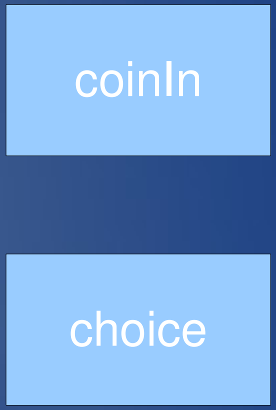
Example: A vending machine

- insert a coin
- make a choice
- confirm choice
    - when you confirm your choice, if a coin is in, a drink pops out

Design the vending machine

- Identify the set of variables that will represent the State of the machine
- Identify the set of functions that will represent the functions available to the user
- Let's assume infinite supply of drinks from the machine
- Let's also assume that the machine has capacity to hold infinite coins

The State



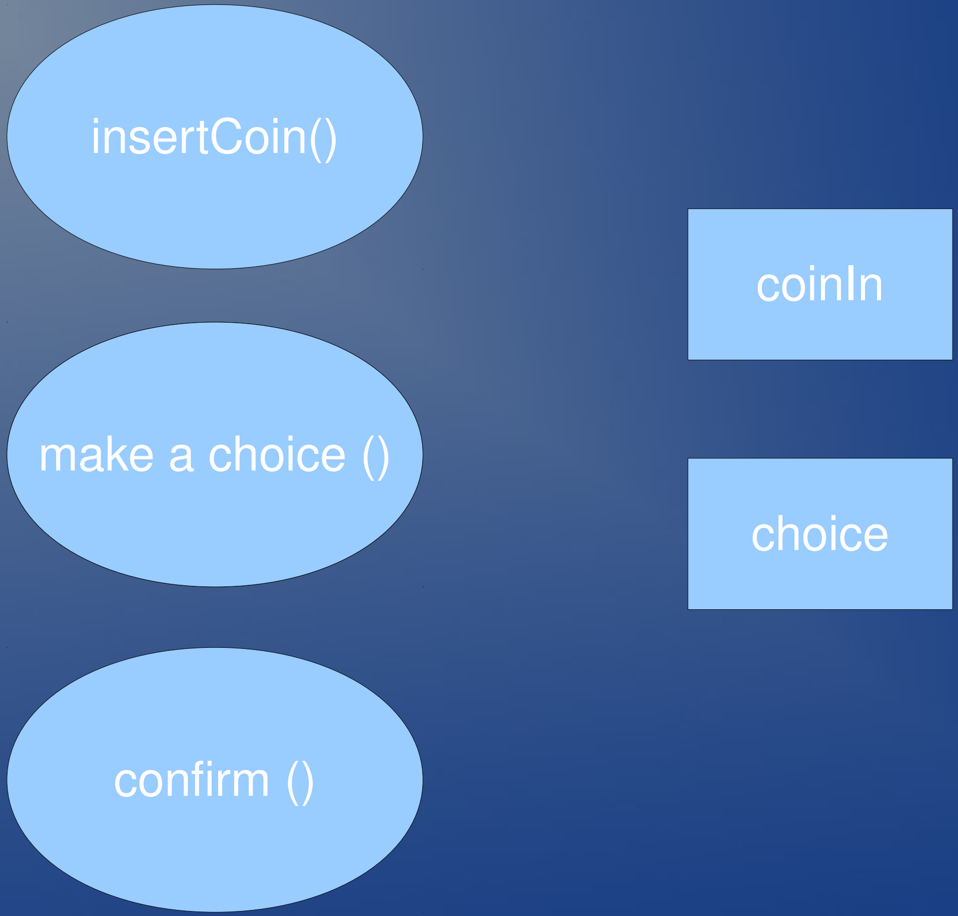
The functions

insertCoin()

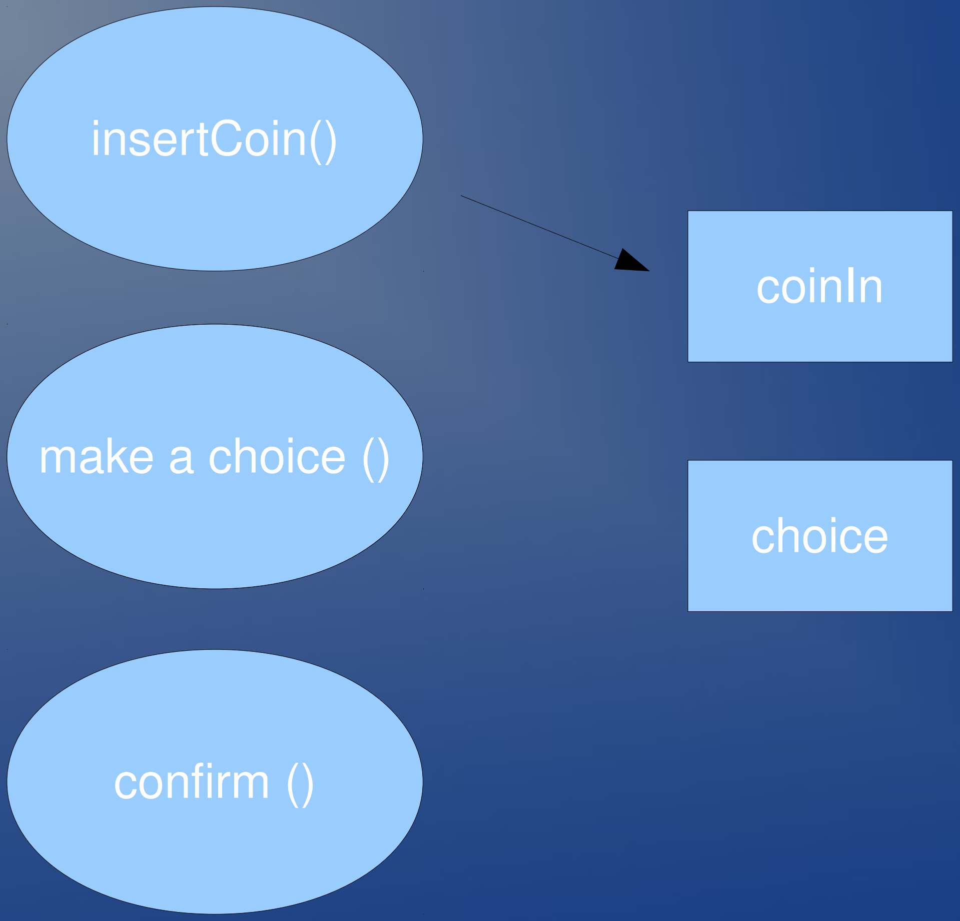
make a choice ()

confirm ()

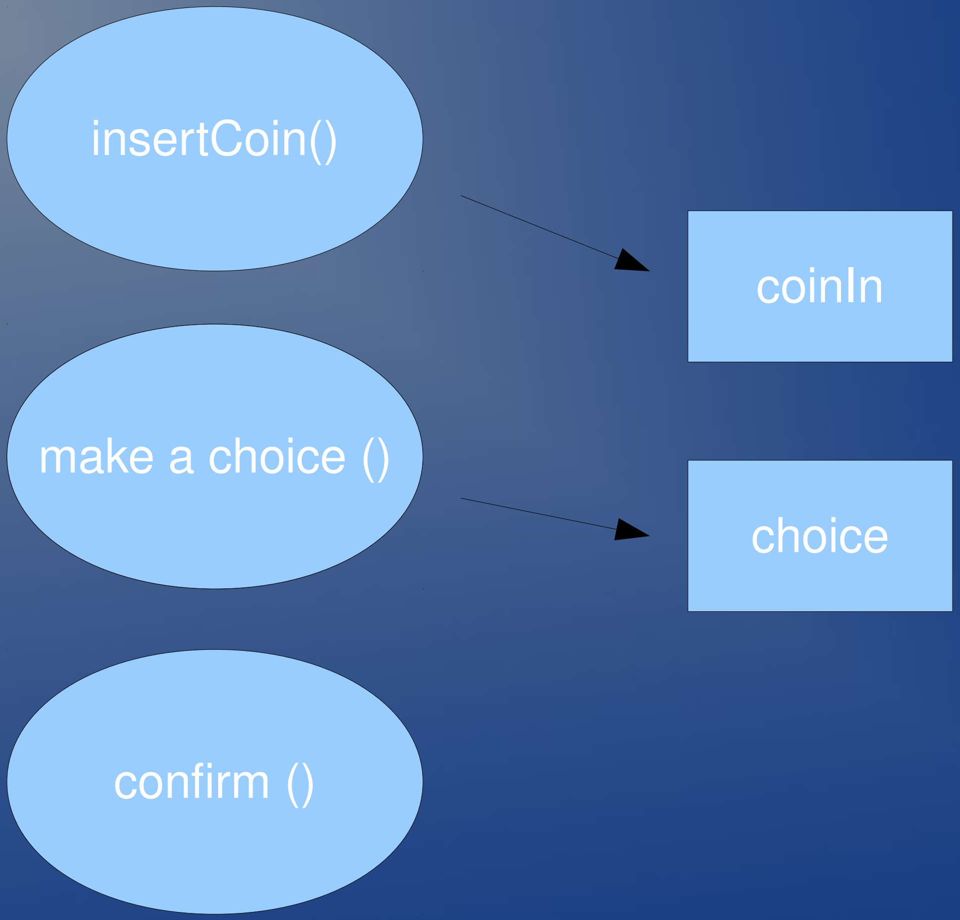
Who accesses what?



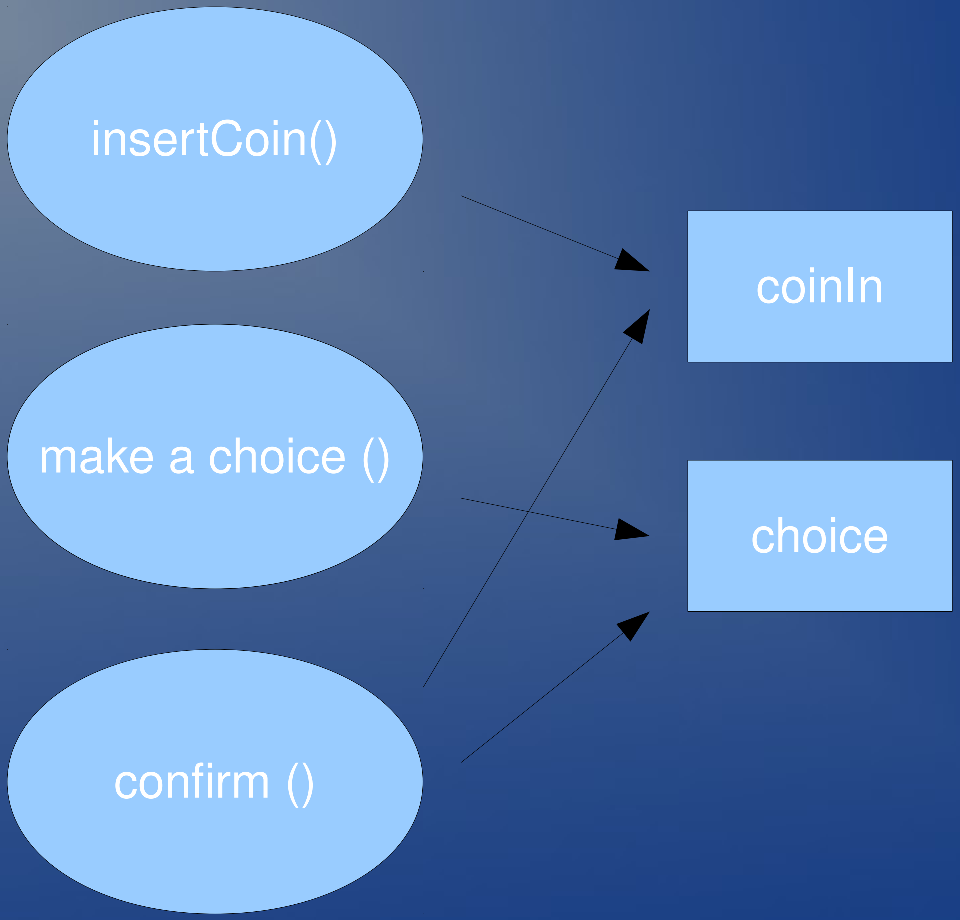
The Accesses



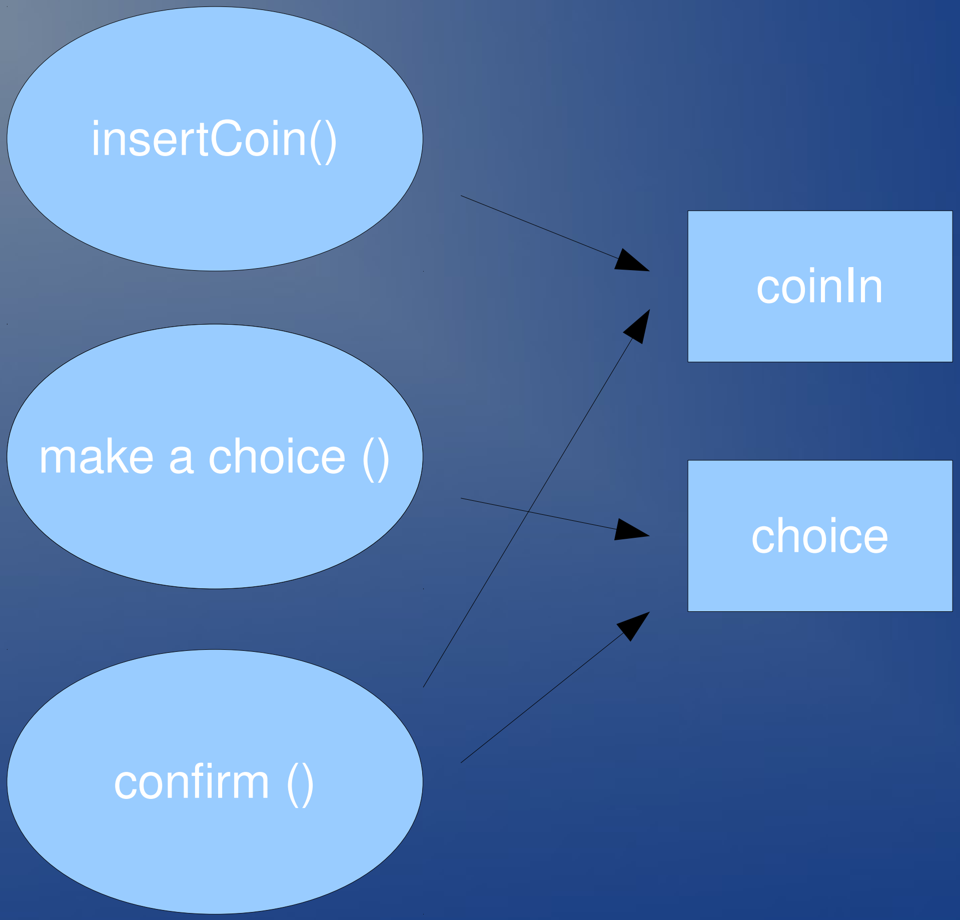
The Accesses



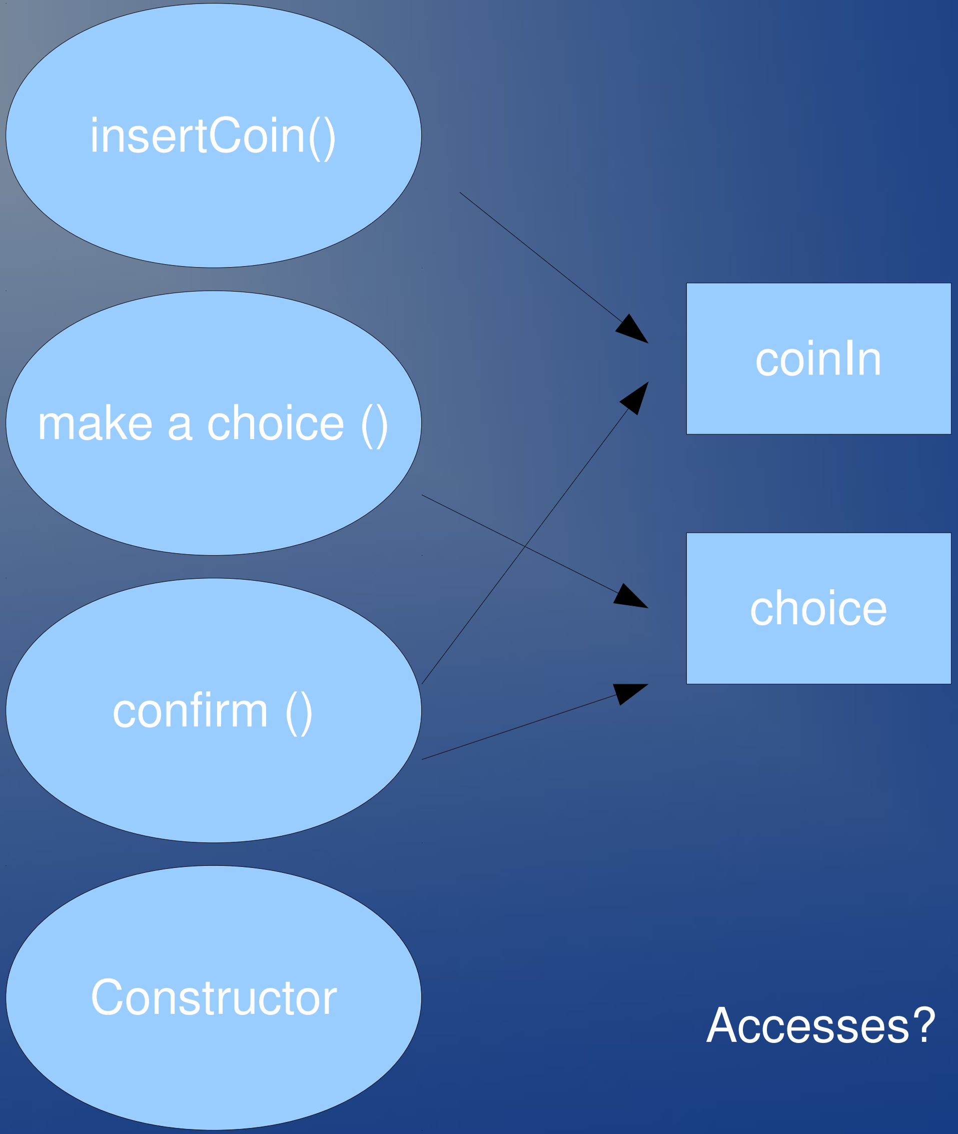
The Accesses



But a component is missing, can you identify it?

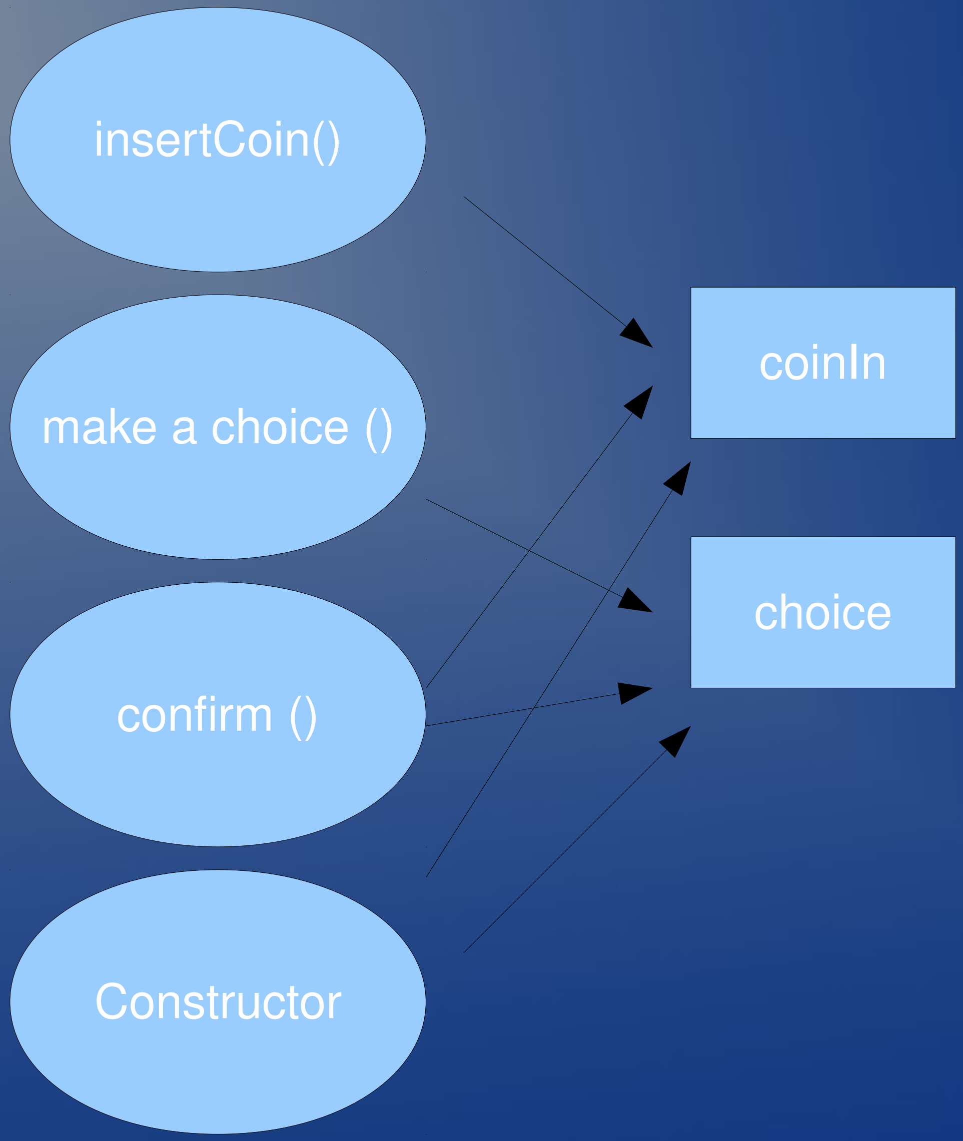


When you construct a vending machine, you will need to initialize the state: Constructor



Accesses?

Constructor: coin is not in, choice is not made!



Some important observations

- The state is shared between these four functions listed on the previous slide
- It is not really a global state for everyone outside the vending machine
- no other function in the program that uses a vending machine should be able to access this state!
  - not even main!
- How to achieve this?!

Exclusive sharing

- We have so far used files to hold together functions, main, and global variables if any
- So, the main and every function in this file can access every other function and global variables
- But we want a more finer control on sharing
- We don't want every function or even the main to see some variables that are to be exclusively shared by some collaborating functions

- *Real life components, equipments are designed with these properties*

A Class

- The 'class' construct can be used to define the behavior of objects such as vending machines
- A class puts everything that we worked out together
  - functions (members of the class)
  - state (shared by member functions)
  - accesses by member functions to state
  - constructor for initializing

A Class

- A class can make some member functions available for public use
- A class also has the property that the state can be concealed inside to be accessed only by the functions that belong to the class
- Private vs. public

We give names to classes

- In our case, we can create a class called
  - class VendingMachine
- It can include the member functions of the vending machine, and the state as identified
- The member functions will access the state
- There has to be a constructor for initialization
- State is private
- Member functions are public

Class vending machine

```
class VendingMachine {




}
```

Class vending machine

```
class VendingMachine {
 private:
     int coin;
     int choice;



}
```

Class vending machine

```
class VendingMachine {
 private:
     int coin;
     int choice;
public:
    VendingMachine();     // constructor!
    void insertCoin ();
     void makeAChoice(unsigned int choice);
     void confirm();
}
```

And the definitions of member functions

```
VendingMachine::VendingMachine() {
     coin=0;
     choice=0;
};
void VendingMachine::insertCoin () {
    coin=1;
};
....
```

Classes provide definitions and objects are the actual values

- Instance (i.e. objects) are created from a class
- int i,j,k;
- VendingMachine v1,v2stat,v3;
- How to invoke functions?
  - v1.insertCoin();
  - v1.confirm();
  - v2.confirm();
- Each object keeps a separate copy of its state