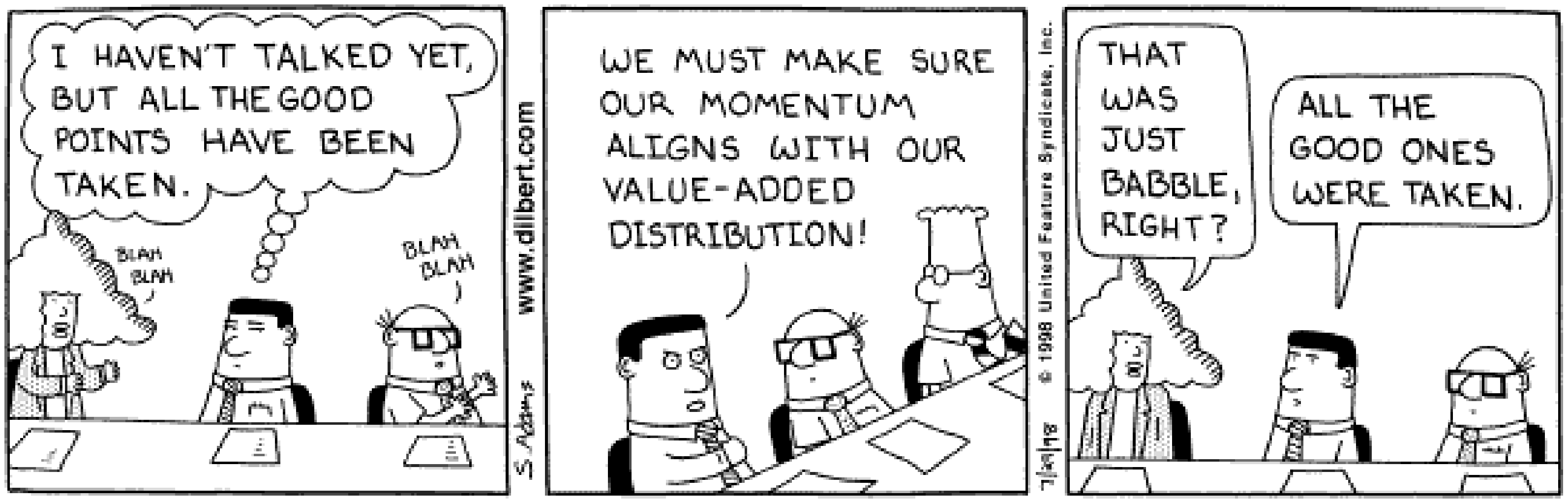
Understanding class definitions

Looking inside classes


I HAVEN'T TALKED YET, BUT ALL THE GOOD POINTS HAVE BEEN TAKEN.
BLAH BLAH
BLAH BLAH
www.dilbert.com
S. Adams
WE MUST MAKE SURE OUR MOMENTUM ALIGNS WITH OUR VALUE-ADDED DISTRIBUTION!
© 1998 United Feature Syndicate, Inc.
THAT WAS JUST BABBLE, RIGHT?
ALL THE GOOD ONES WERE TAKEN.

Time for study

- A full-time student week is 40 hours!

Examination

- Course work: 3 assignments
- Programming test
  - in lab; at computer
  - practical task
  - exam conditions
- Programming test <u>MUST</u> be passed
  - (pass/fail mark; hurdle requirement)
- Final mark calculated from coursework marks

Why BlueJ

- Why Java?
- Why BlueJ?

And, by the way:

- Greenfoot

Main concepts to be covered

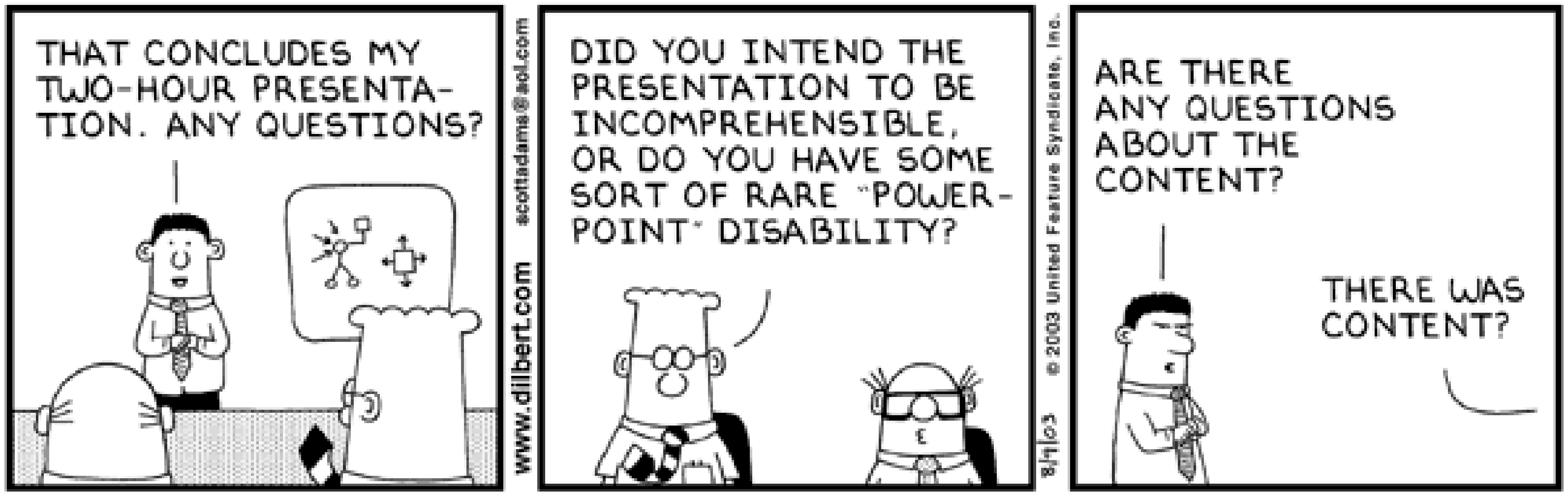
- fields
- constructors
- methods
- parameters
- assignment statements

Ticket machines

Demo

Ticket machines – an internal view

- Interacting with an object gives us clues about its behaviour.
- Looking inside allows us to determine how that behaviour is provided or implemented.
- All Java classes have a similar-looking internal view.


THAT CONCLUDES MY TWO-HOUR PRESENTATION. ANY QUESTIONS?
www.dilbert.com scottadams@aol.com
DID YOU INTEND THE PRESENTATION TO BE INCOMPREHENSIBLE, OR DO YOU HAVE SOME SORT OF RARE "POWERPOINT" DISABILITY?
8/4/03 © 2003 United Feature Syndicate, Inc.
ARE THERE ANY QUESTIONS ABOUT THE CONTENT?
THERE WAS CONTENT?

Basic class structure

```
public class TicketMachine
{
    Inner part of the class omitted.
}
```

The outer wrapper of TicketMachine

```
public class ClassName
{
    Fields
    Constructors
    Methods
}
```

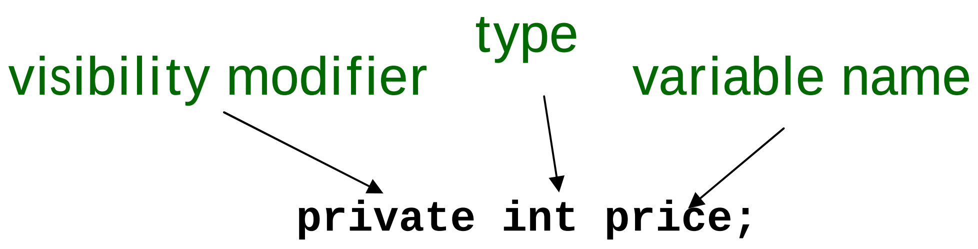
The contents of a class

Fields

- Fields store values for an object.
- They are also known as instance variables.
- Use the *Inspect* option to view an object's fields.
- Fields define the state of an object.

```
public class TicketMachine
{
    private int price;
    private int balance;
    private int total;

    Further details omitted.
}
```



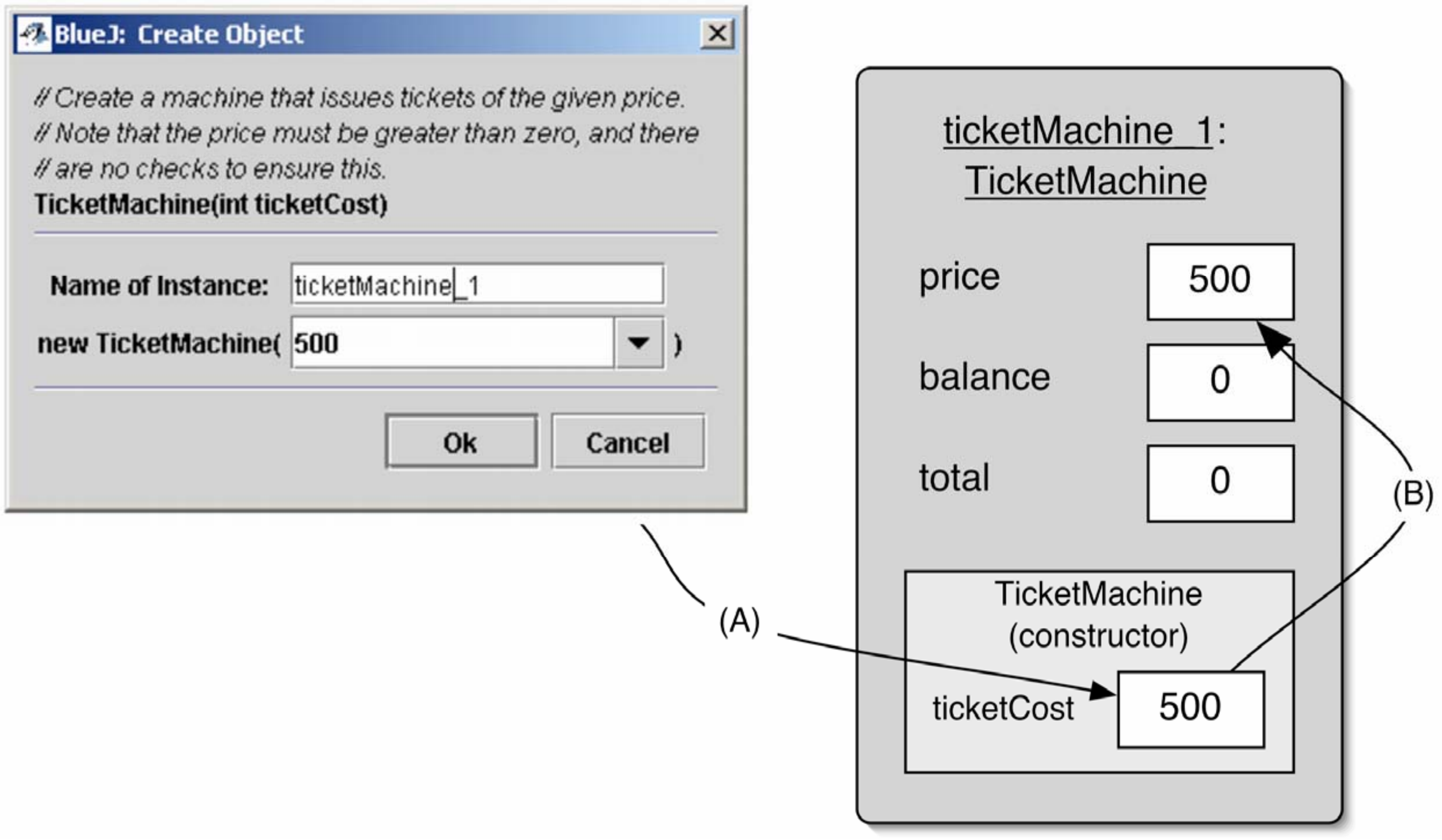
```
private int price;
```

Constructors

- Constructors initialise an object.
- They have the same name as their class.
- They store initial values into the fields.
- They often receive external parameter values for this.

```
public TicketMachine(int ticketCost)
{
    price = ticketCost;
    balance = 0;
    total = 0;
}
```

Passing data via parameters



// Create a machine that issues tickets of the given price.
// Note that the price must be greater than zero, and there
// are no checks to ensure this.
TicketMachine(int ticketCost)

Name of Instance: ticketMachine_1
new TicketMachine(500)

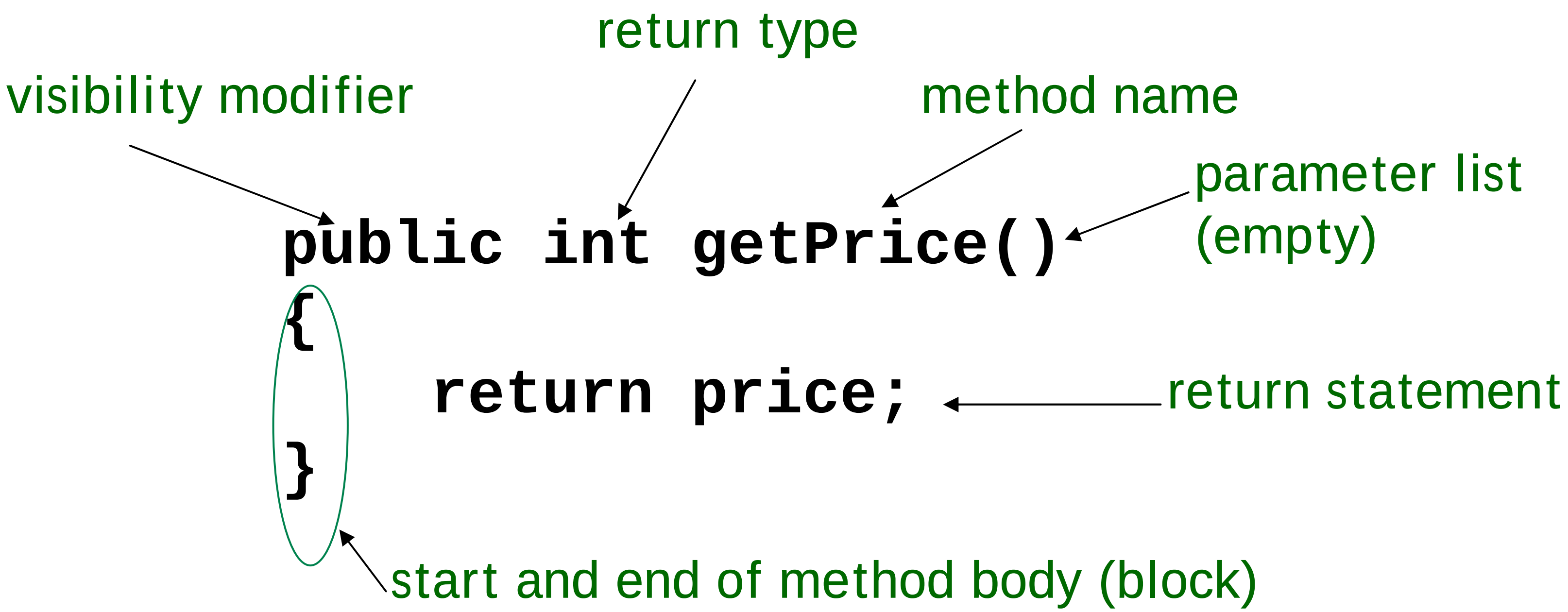
Assignment

- Values are stored into fields (and other variables) via assignment statements:
  - *variable = expression;*
  - `price = ticketCost;`
- A variable stores a single value, so any previous value is lost.

Accessor methods

- Methods implement the behaviour of objects.
- Accessors provide information about an object.
- Methods have a structure consisting of a header and a body.
- The header defines the method’s *signature*.
  - `public int getPrice()`
- The body encloses the method’s statements.

Accessor methods



```
public int getPrice()
{
    return price;
}
```

Test

```
public class CokeMachine
{
  private price;

  public CokeMachine()
  {
     price = 300
  }

  public int getPrice
  {
     return Price;
  }
```

- What is wrong here?

(there are five errors!)

Test

```
public class CokeMachine
{
            int
   private price;

   public CokeMachine()
   {
      price = 300;
   }

   public int getPrice()
   {
             p
      return Price;
   }
}
```

- What is wrong here?

(there are <u>five</u> errors!)

Mutator methods

- Have a similar method structure: header and body.
- Used to *mutate* (i.e. change) an object's state.
- Achieved through changing the value of one or more fields.
    - Typically contain assignment statements.
    - Typically receive parameters.

Mutator methods

visibility modifier

return type

method name

parameter

```
public void insertMoney(int amount)
{
    balance = balance + amount;
}
```

field being mutated

assignment statement

Printing from methods

```
public void printTicket()
{
    // Simulate the printing of a ticket.
    System.out.println("##################");
    System.out.println("# The BlueJ Line");
    System.out.println("# Ticket");
    System.out.println("# " + price + " cents.");
    System.out.println("##################");
    System.out.println();

    // Update the total collected with the balance.
    total = total + balance;
    // Clear the balance.
    balance = 0;
}
```

Reflecting on the ticket machines

- Their behaviour is inadequate in several ways:
  - No checks on the amounts entered.
  - No refunds.
  - No checks for a sensible initialisation.
- How can we do better?
  - We need more sophisticated behaviour.