Week 3

Understanding class definitions

Looking inside classes

suggested reading:

Textbook, Ch. 2

Main concepts to be covered

- fields
- constructors
- methods
- parameters
- statements
  - printing, conditional, and assigment statements

Ticket machines

Demo

Example: project "ticket machine"

- one class only: TicketMachine
- What's it all about?
  - machine sells one type of tickets
  - one price
  - collects the correct amount of money.

Ticket machines – an external view

- Exploring the behaviour of a typical ticket machine.
  - Machines supply tickets of a fixed price - how is that price determined?
  - How is ‘money’ entered into a machine?
  - How does a machine keep track of the money that has been entered so far?

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.

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

(order of appearance is a convention, but not mandatory)

Fields,constructors, & methods

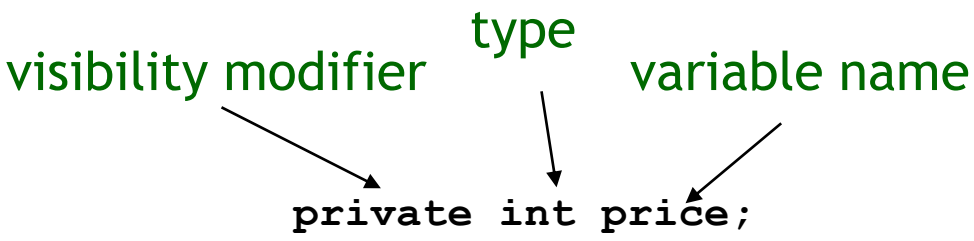
- Fields characterize an object, i.e. contain specific values.
- Constructors initialize objects by assigning initial values to their fields.
- Methods implement the behavior of objects -> any kind of further action after the initialization.

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.
}
```



Attention: Comments!

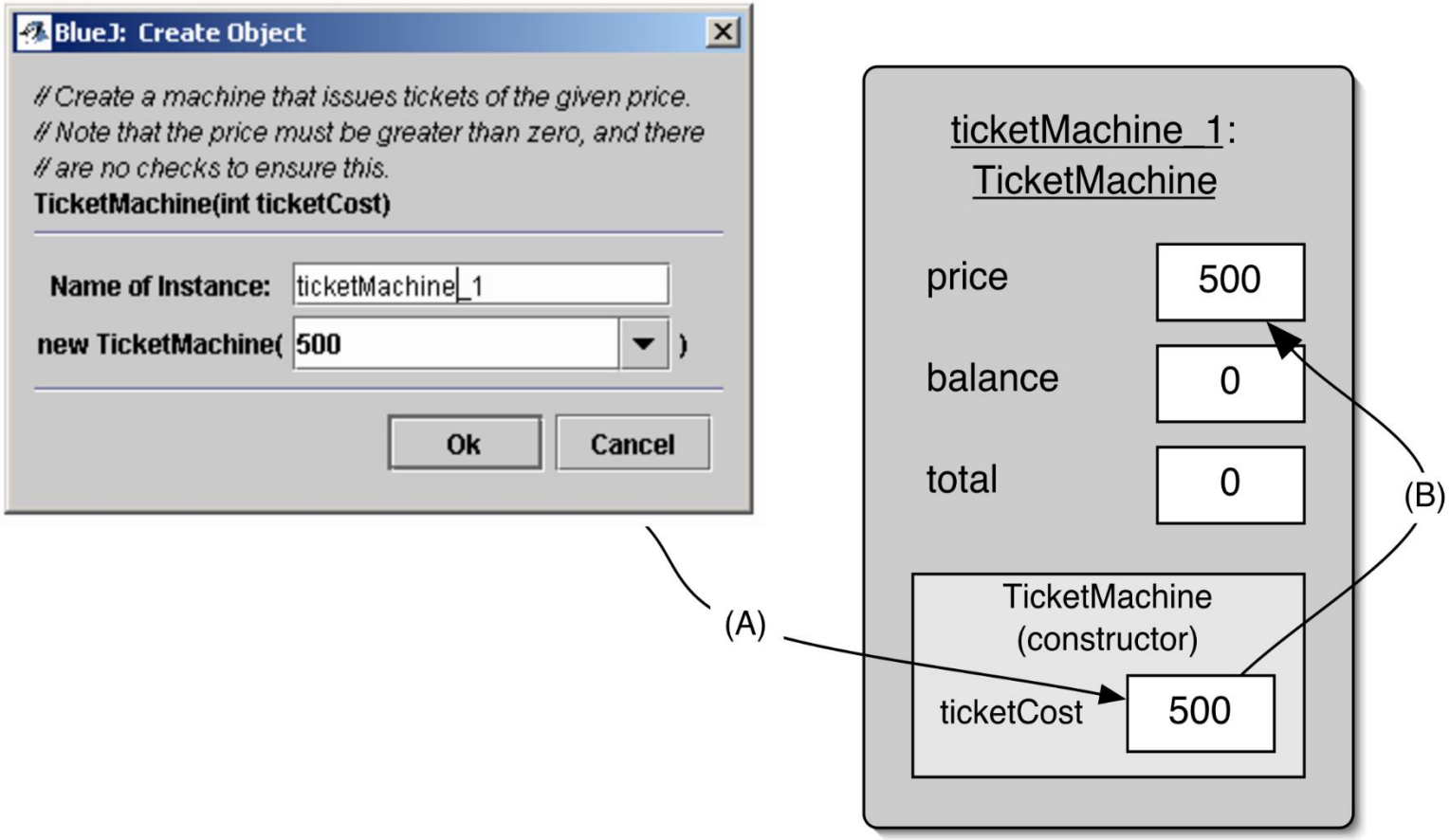
- Comments are important for code documentation.
- Adding comments is a question of good programming style.
- One-line comments: `//` …
- Multi-line comments: `/*` … `*/`
- (javadoc comments: `/**` … `*/`)

Constructors

- Constructors initialize 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



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.
- Assignment operators: `=`, `+=`, `-=`

Methods

- Our `TicketMachine` has four methods:
  - `getPrice`
  - `getBalance`
  - `insertMoney`
  - `printTicket`

Accessor methods

- Methods implement the behavior 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

() makes it a method!

return type

visibility modifier

method name

parameter list (empty)

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

return statement

start and end of method body (block)

Test

```
public class CokeMachine
{
  private int price;

  public CokeMachine()
  {
     price = 300;
  }

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

- What is wrong here?

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

Test

```
public class CokeMachine
{
    int
    private price;

    public CokeMachine()
    {
        price = 300;
    }

    public int getPrice()
    {
        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

Note that constructors have no return type!!

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.

Making choices – conditional statements

'if' keyword

boolean condition to be tested

actions if condition is true

```
if (perform some test) {
    Do these statements if the test gives true
}
else {
    Do these statements if the test gives false
}
```

'else' keyword

actions if condition is false

Making choices – conditional statements

```
public void insertMoney(int amount)
{
    if(amount > 0) {
        balance = balance + amount;
    }
    else {
        System.out.println("Use a positive amount: "
                                + amount);
    }
}
```

Local variables

- Fields are one sort of variable.
  - They store values through the life of an object.
  - They are accessible throughout the class.
- Methods can include shorter-lived variables.
  - They exist only as long as the method is being executed.
  - They are only accessible from within the method.

Scope and life time

- The scope of a local variable is the block it is declared in.
- The lifetime of a local variable is the time of execution of the block it is declared in.

Local variables

A local variable, not a field (fields always defined outside methods!)

No visibility modifier

```
public int refundBalance()
{
    int amountToRefund;
    amountToRefund = balance;
    balance = 0;
    return amountToRefund;
}
```

- local variables also possible in constructors!

Review (I)

- Class bodies contain fields, constructors and methods.
- Fields store values that determine an object's state.
- Constructors initialize objects.
- Methods implement the behavior of objects.

Review (II)

- Fields, parameters and local variables are all variables.
- Fields persist for the lifetime of an object.
- Parameters are used to receive values into a constructor or method.
- Local variables are used for short-lived temporary storage.

Review (III)

- Objects can make decisions via conditional (if) statements.
- A true or false test allows one of two alternative courses of actions to be taken.