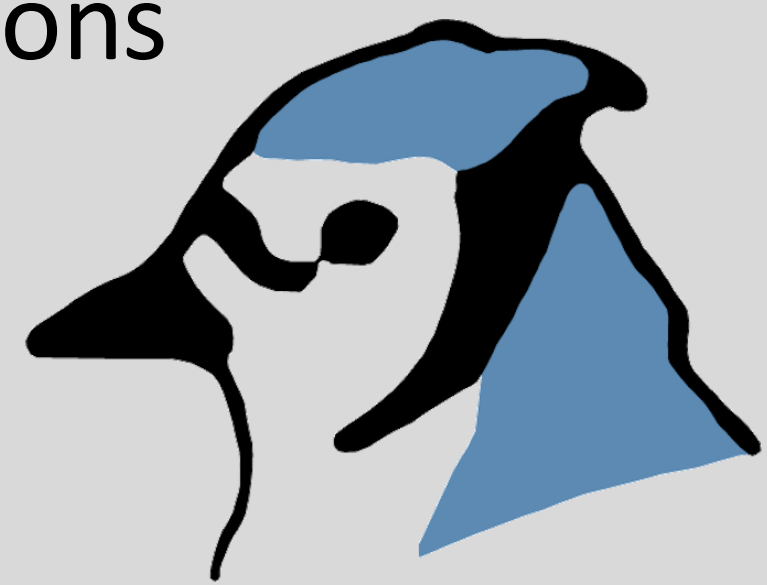
Understanding class definitions

Exploring source code



Produced by: Ms. Mairead Meagher,
Ms. Siobhán Roche.

Ollscoil Teicneolaíochta an Oirdheiscirt
South East Technological University

Department of Computing and Mathematics
http://www.setu.ie/

Main concepts to be covered

- fields
- constructors
- methods
- parameters
- assignment statements

Ticket machine Example

- This is a very simple Ticket machine.
- It prints a ticket when an amount of money is entered

Ticket machines – an external view

- Exploring the behavior of a typical ticket machine.
  - Use the *naive-ticket-machine* project.
  - 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 is entered?

Ticket machine Example

What information do we need to store?

What can the Ticket Machine do?

Ticket machine Example

What information do we need to store?

- Price of ticket
- Amount entered
- Amount in machine
- …

What can the Ticket Machine do?

- Accept Payment
- Print Ticket
- …

Ticket machine Example

fields

What information do we need to store?

- Price of ticket
- Number of tickets sold
- Amount in machine
- ?

methods

What can the Ticket Machine do?

- Accept Payment
- Print Ticket
- ?

Ticket machines – an internal view

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

Basic class structure

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

The outer wrapper of TicketMachine

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

The inner contents of a class

Keywords

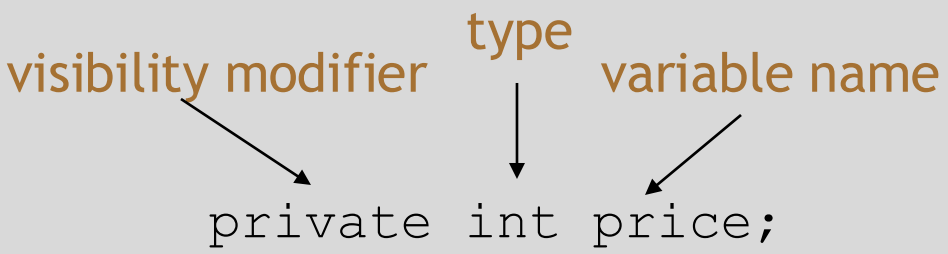
- Words with a special meaning in the language:
  - `public`
  - `class`
  - `private`
  - `int`
- Also known as *reserved words*.
- Always entirely lower-case.

Fields

- Fields store values for an object.
- They are also known as *instance variables*.
- Fields define the state of an object.
- Use *Inspect* to view the state.
- Some values change often.
- Some change rarely (or not at all).

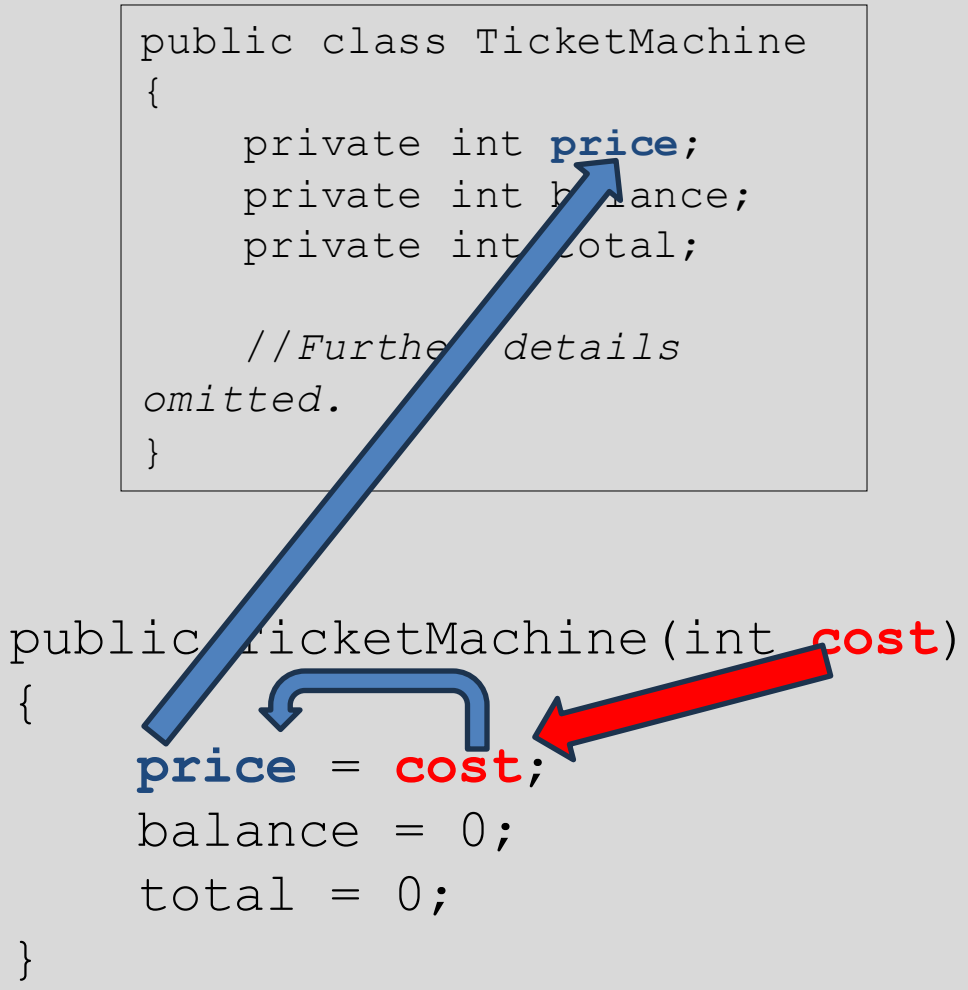
```
public class TicketMachine
{
    private int price;
    private int balance;
    private int total;

    //Further details omitted.
}
```

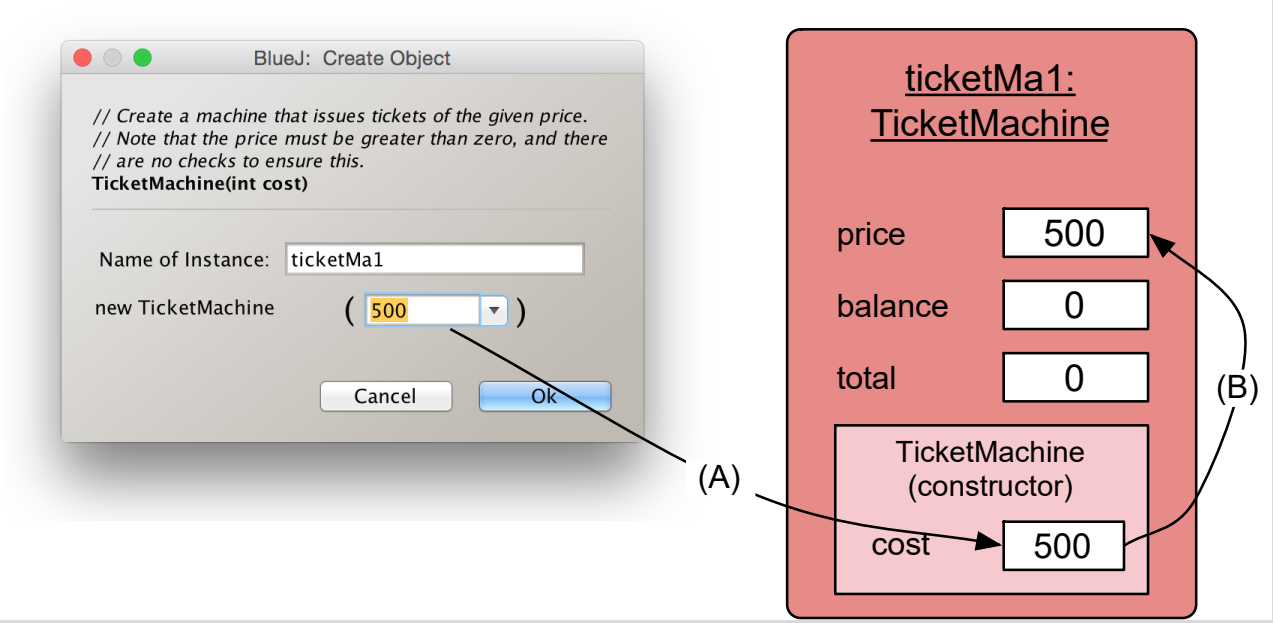


Constructors

- Initialise an object.
- Have the same name as their class.
- Close association with the fields:
  - Initial values stored into the fields.
  - Parameter values often used for these.



Passing data via parameters



Parameters are another sort of variable.

Variables

In Programming, variables:

- are created (defined) in your programs
- are used to store data (whose value can change over time)
- have a data type
- have a name
- are a VERY important programming concept

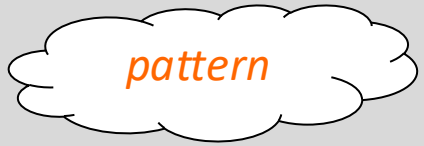
Choosing variable names

- There is a lot of freedom over choice of names. Use it wisely!
- Choose expressive names to make code easier to understand:
  - **`price`**, **`amount`**, **`name`**, **`age`**, etc.
- Avoid single-letter or cryptic names:
  - **`w`**, **`t5`**, **`xyz123`**

A good variable name is worth a lot of documentation

Assignment

- Values are stored into fields (and other variables) via assignment statements:
  - *variable = expression;*



  - **`balance = balance + amount;`**

- A variable can store just one value, so any previous value is lost.

- Assignment statements work by
  - Eavluating what appears on the right-hand side of the operator and
  - copying that value into a variable on the left-hand side.

Next concepts to be covered

- Methods, including:
  - *accessor* (*getter*) methods
  - *mutator* (*setter*) methods;
- String formatting;
- Conditional statements;
- Local variables.

Questions?