

Today's Agenda

- TA Information
- Homework 1, Due on 6/17
- Quick Review
- Finish **Objects and Classes**
- **Understanding class definitions**

Quick Review

- What is OOP? How is OOP different from procedural programming?
- What is an object? What is a class?

Understanding class definitions

Looking inside classes

4.0

Main concepts to be covered

- fields
- constructors
- methods
- parameters
- assignment statements

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.
 - Methods *insertMoney*, *getBalance*, and *printTicket* are used to enter money, keep track of balance, and print out tickets.

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 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.
}
```

visibility modifier type variable name

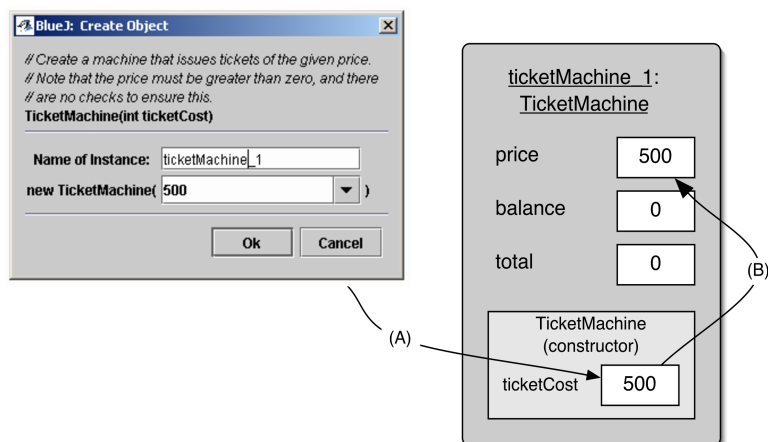
private int price;

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.

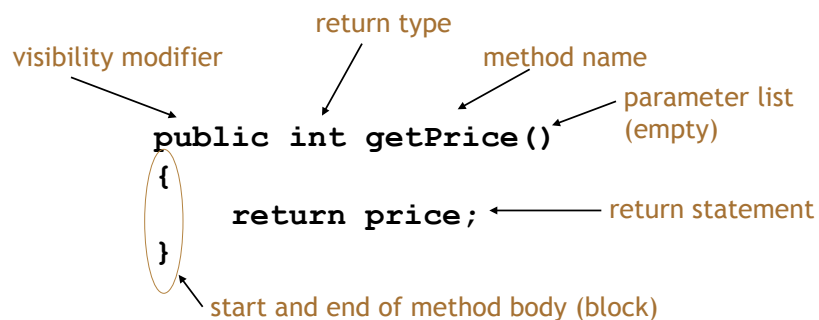
Main concepts to be covered

- mutator and accessor methods
- conditional statements
- local variables
- string concatenation

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



Test

```
public class CokeMachine
{
    private price;

    public CokeMachine()
    {
        price = 300
    }

    public int getPrice
    {
        return Price;
    }
}
```

- What is wrong here?

(there are five errors!)

CSE 1325: Object-Oriented Programming in Java

15

Test

```
public class CokeMachine
{
    int private price;

    public CokeMachine()
    {
        price = 300;
    }

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

- What is wrong here?

(there are five errors!)

CSE 1325: Object-Oriented Programming in Java

16

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;
}
```

CSE 1325: Object-Oriented Programming in Java

19

String concatenation

- $4 + 5$
9 → overloading
- "wind" + "ow"
"window"
- "Result: " + 6
"Result: 6"
- "# " + price + " cents"
"# 500 cents"

CSE 1325: Object-Oriented Programming in Java

20

Quiz

- `System.out.println(5 + 6 + "hello");`

`11hello`

- `System.out.println("hello" + 5 + 6);`

`hello56`

Reflecting on the ticket machines

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

Making choices

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

Making choices

'if' keyword

boolean condition to be tested

actions if condition is true

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

'else' keyword

actions if condition is false

How do we write 'refundBalance'?

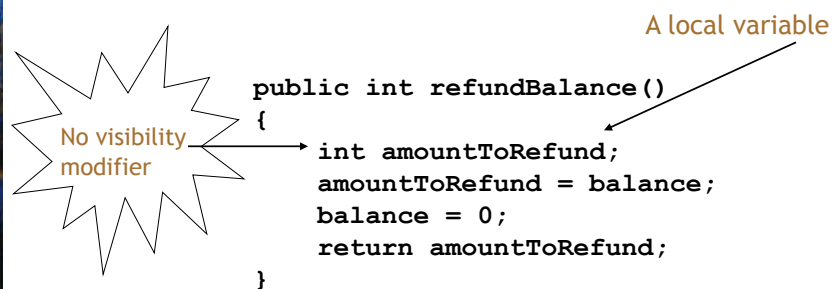
Local variables

- 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



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

Review

- 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

- 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

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