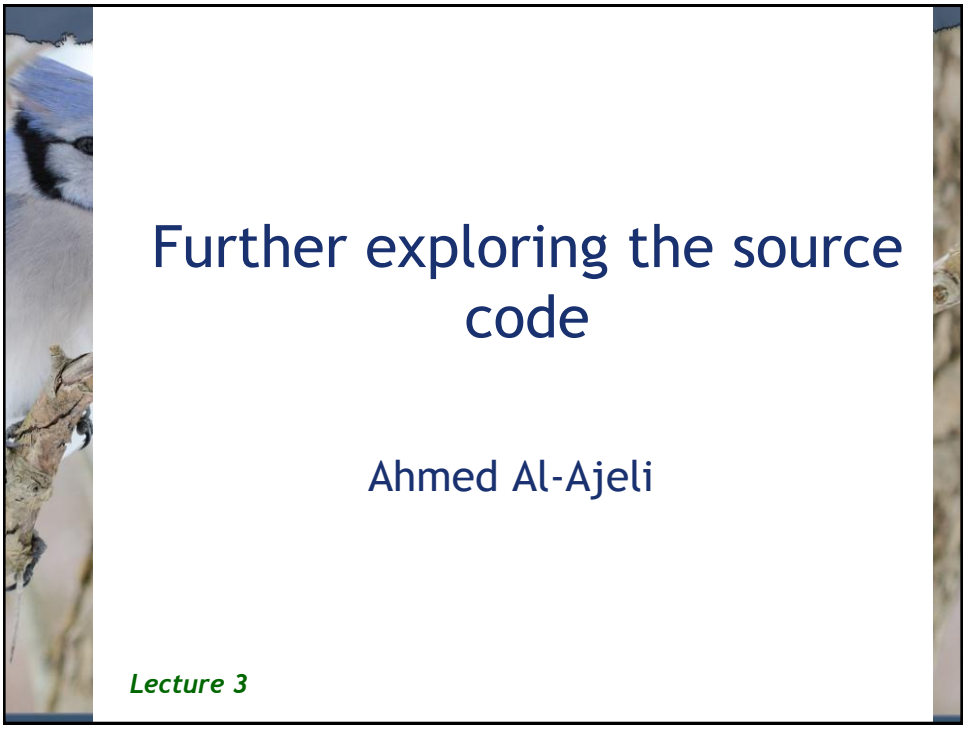
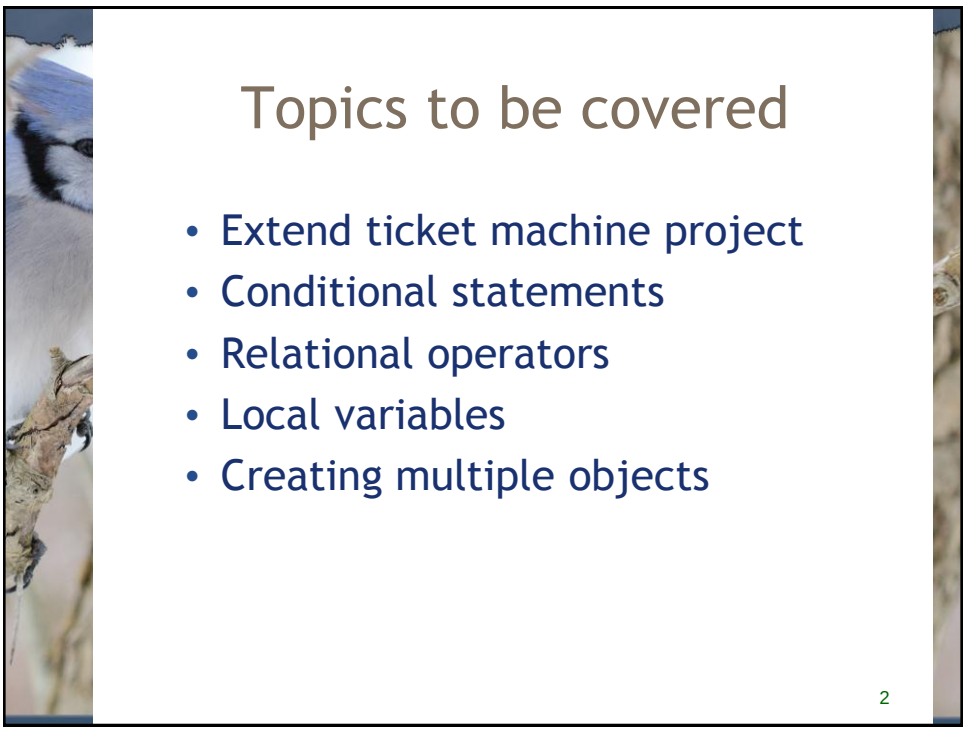


Further exploring the source code

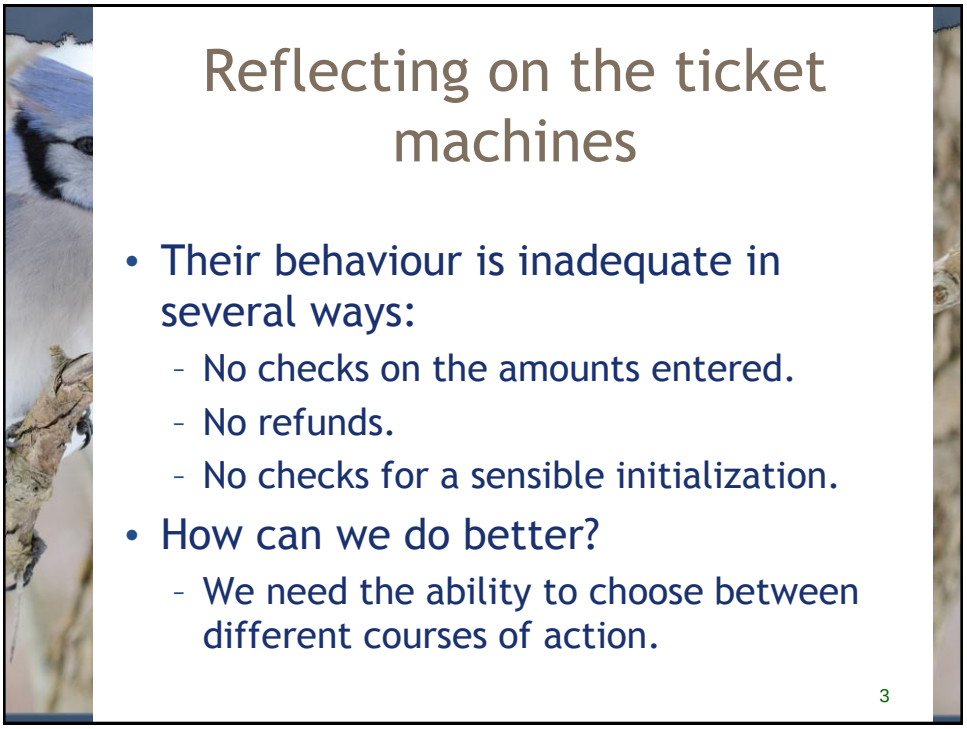
Ahmed Al-Ajeli

Lecture 3



Topics to be covered

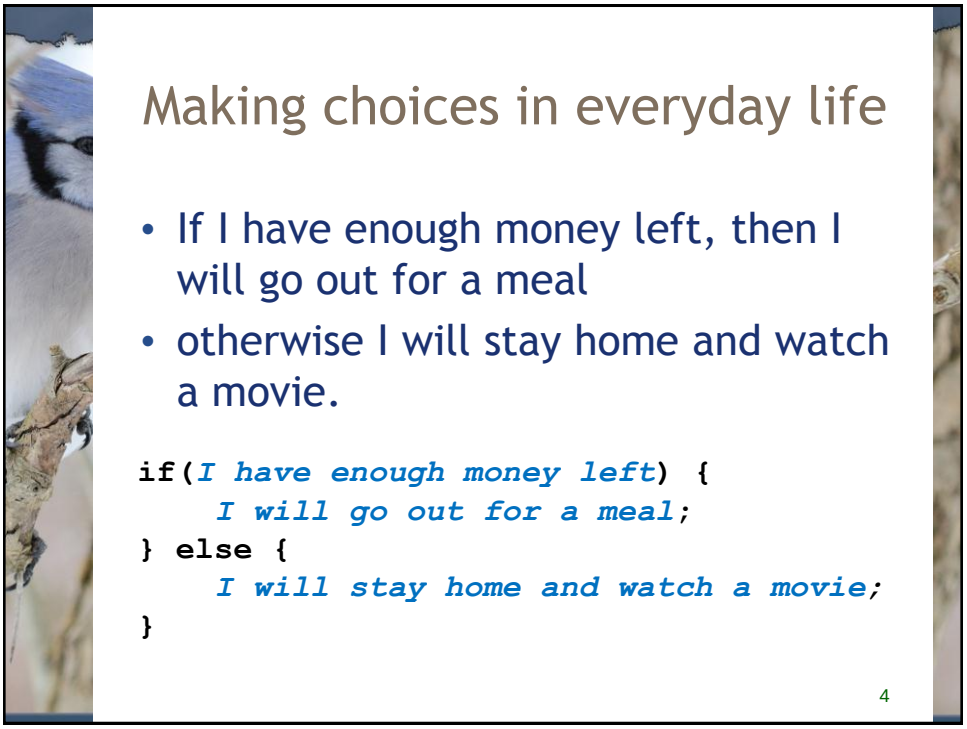
- Extend ticket machine project
- Conditional statements
- Relational operators
- Local variables
- Creating multiple objects



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 initialization.
- How can we do better?
  - We need the ability to choose between different courses of action.

3



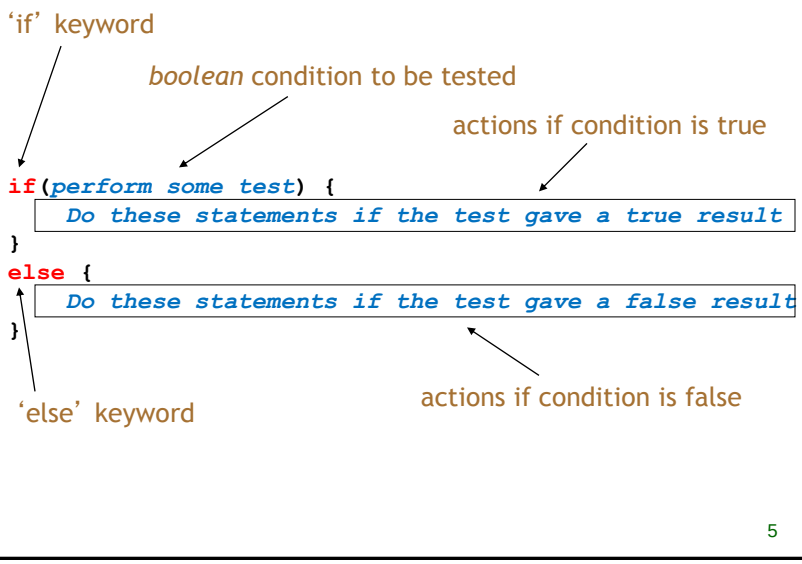
Making choices in everyday life

- If I have enough money left, then I will go out for a meal
- otherwise I will stay home and watch a movie.

```
if(I have enough money left) {
    I will go out for a meal;
} else {
    I will stay home and watch a movie;
}
```

4

Making choices in Java



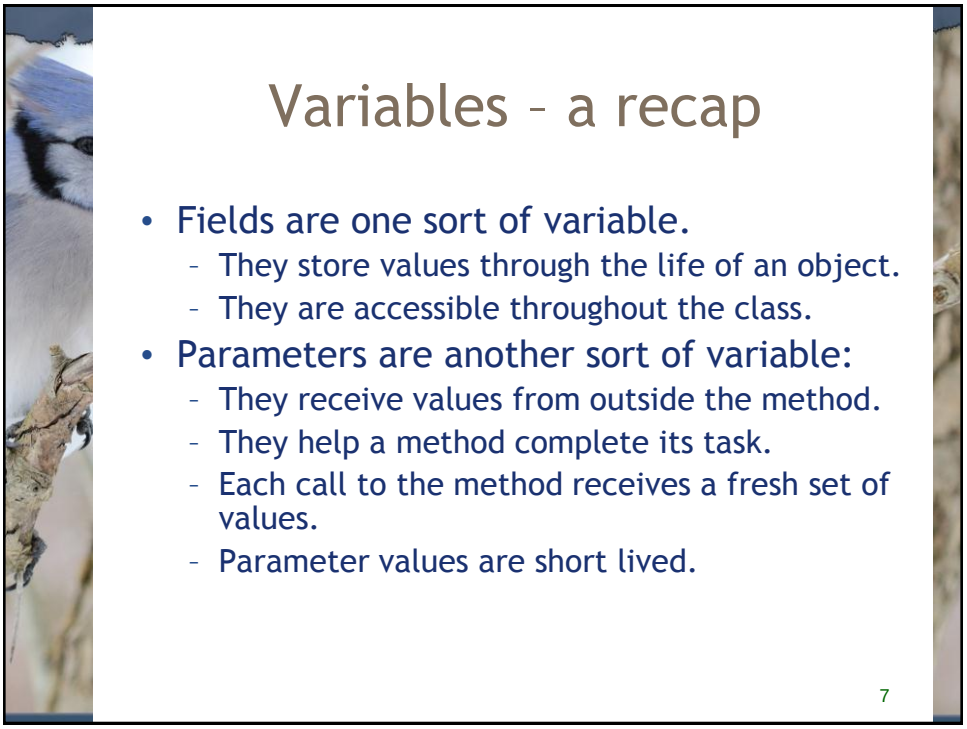
5

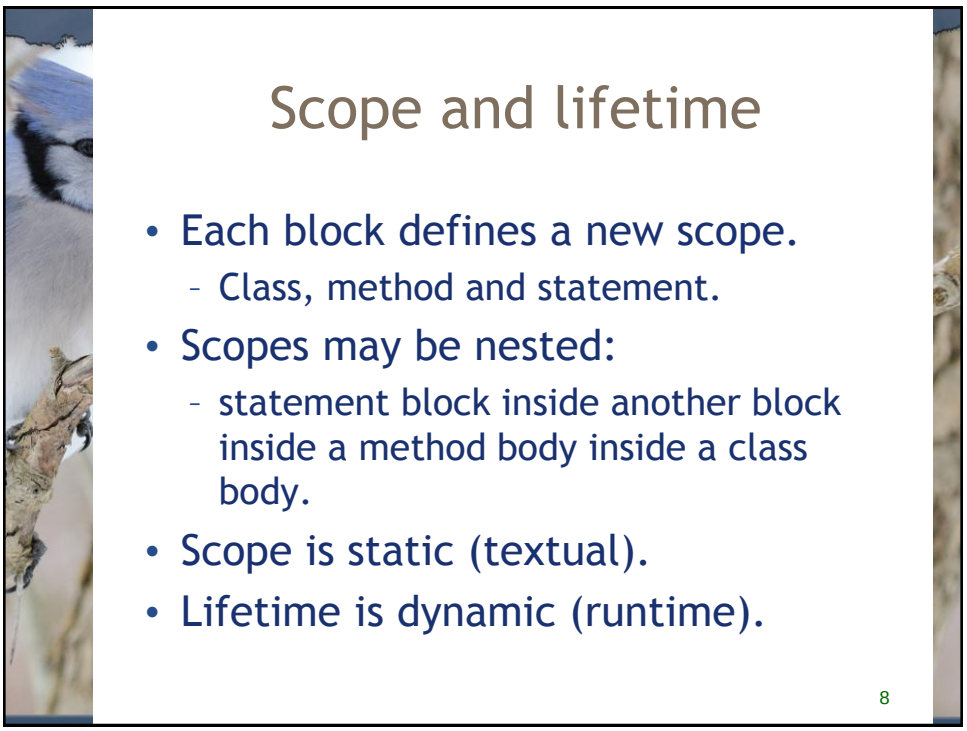
Making a choice in the ticket machine

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

conditional statement avoids an inappropriate action

6


Variables – a recap
• Fields are one sort of variable.
– They store values through the life of an object.
– They are accessible throughout the class.
• Parameters are another sort of variable:
– They receive values from outside the method.
– They help a method complete its task.
– Each call to the method receives a fresh set of values.
– Parameter values are short lived.
7


Scope and lifetime
• Each block defines a new scope.
– Class, method and statement.
• Scopes may be nested:
– statement block inside another block inside a method body inside a class body.
• Scope is static (textual).
• Lifetime is dynamic (runtime).
8

printTicket method

```
public void printTicket()
{
   if(balance >= price) {
       // Simulate the printing of a ticket.
       System.out.println("###################");
       System.out.println("# The Eclipse Line");
       System.out.println("# Ticket");
       System.out.println("# " + price + " cents.");
       System.out.println("###################");
       System.out.println();
       // Update the total collected with the price.
       total = total + price;
       // Reduce the balance by the price.
       balance = balance - price;
   }
   else {
       System.out.println("You must insert at least: " +
       (price - balance) + " cents.");
   }
}
```

9

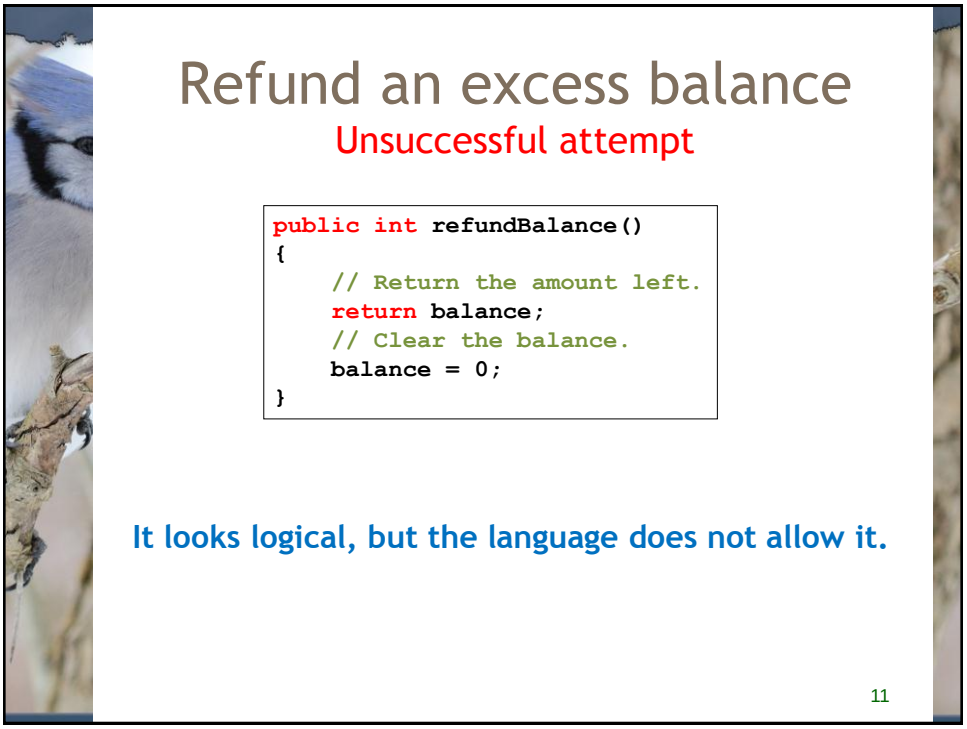
Relational operators

Standard relational operators are provided for the *if* test

- < less than
- > greater than
- <= less than or equal to
- >= greater than or equal to
- == equal to
- != not equal to

a comparison using a relational operator is known as a *Boolean expression*, since it evaluates to a *Boolean* (true or false) value

10



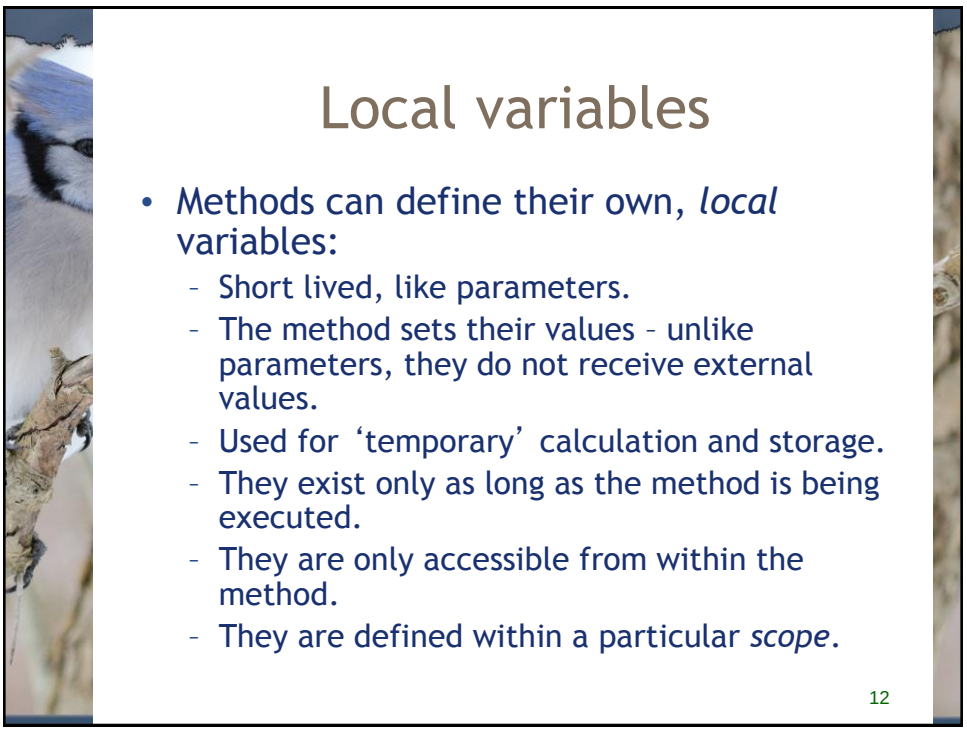
Refund an excess balance

Unsuccessful attempt

```
public int refundBalance()
{
    // Return the amount left.
    return balance;
    // Clear the balance.
    balance = 0;
}
```

It looks logical, but the language does not allow it.

11

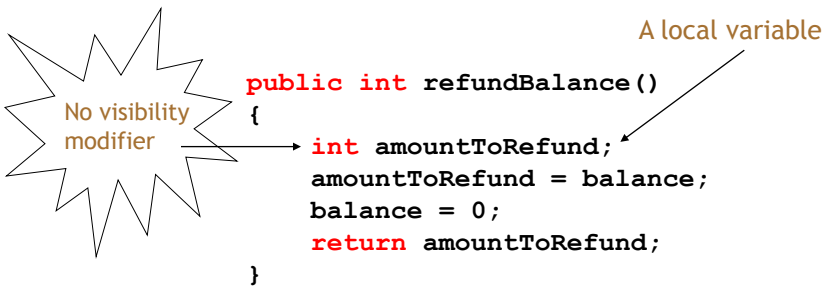


Local variables

- Methods can define their own, *local* variables:
  - Short lived, like parameters.
  - The method sets their values – unlike parameters, they do not receive external values.
  - Used for 'temporary' calculation and storage.
  - They exist only as long as the method is being executed.
  - They are only accessible from within the method.
  - They are defined within a particular *scope*.

12

Local variables



you can declare and assign a local variable at the same time

```
int amountToRefund;

amountToRefund = balance;
```

```
int amountToRefund = balance;
```

13

Scope and lifetime

- The scope of a field is its whole class.
- The lifetime of a field is the lifetime of its containing object.
- The scope of a local variable is the block in which it is declared.
- The lifetime of a local variable is the time of execution of the block in which it is declared.

14

Creating multiple objects (1)

We can create multiple instances of the same class using the following pattern:

```
Classname  objname_1, objname_2, …, objname_n;

objname_1 = new Classname (parameters list_1);
objname_2 = new Classname (parameters list_2);
            .
            .
            .
Objname_n = new Classname (parameters list_n);
```

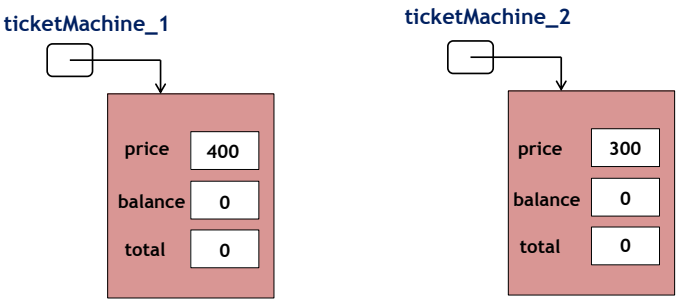
15

Creating multiple objects (2)

Example:

```
TicketMachine ticketMachine_1, ticketMachine_2;

ticketMachine_1 = new TicketMachine (400);
ticketMachine_2 = new TicketMachine (300);
```



16

Reviewing a familiar example

```
public class Student
{
    private String name;
    private String id;
    private int credits;
    public Student(String fullName, String studentID)
    {
        name = fullName;
        id = studentID;
        credits = 0;
    }

    public String getName()
    {
        return name;
    }
```

17

```
    public void changeName(String newName)
    {
        name = newName;
    }

    public String getStudentID()
    {
        return id;
    }
    public void addCredits(int newCreditPoints)
    {
        credits += newCreditPoints;
    }

    public int getCredits()
    {
        return credits;
    }
    public void print()
    {
        System.out.println(name + ", student ID: " + id +
                           ", credits: " + credits);
    }
}
```

18