Objects and Classes

Software Development 1 (F27SA1)

Week 7, lecture 1

*Multiple slides over the course adapted form Verena Rieser @HWU

Outline

- Objects, classes, instances.
- Understanding class definitions.
  - fields
  - constructors
  - assignments

Programming paradigms

- Object Oriented
  - Java, Smalltalk, C++, C#, Python, Ruby,..
- Procedural
  - C, Fortran, Basic, …
- Declarative
  - Prolog, SQL,…
- Functional
  - Lips, Haskell,…
- Etc.

NB: Multi-paradigm Languages

- Most of the “modern” programming languages support more than one paradigm.
- 3 paradigms: **Perl, Tcl, JavaScript**
- 4 paradigms: **Java, C++, Python**
- 5 paradigms: **C♯**

What is an object?

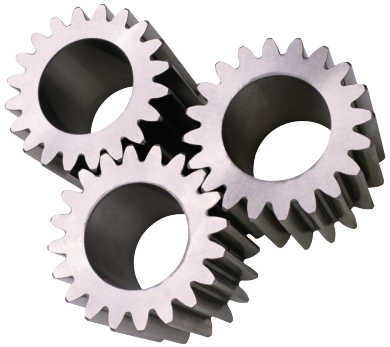
- Name some examples for objects.
  - Bottle, shoe, book, a page,…
  - In Java, also freedom, tomorrow, a message,… are Objects!
- What are other words for “object”?
  - Thing, entity, physical body, abstract concept, sentence object (grammar),…
- What constitutes an object?
  - Has properties, can interact, can be manipulated, some can act themselves, opposite of subject,…

An offline Class and many runtime Objects

Plato's theory of forms/ "Platonic ideas"

Objects, Classes, Instances

Real world → Idealistic blueprint → Real world simulation



OBJECT FACTORY

Real **Object** → Java **Class** → Multiple software object **instances**

Objects, Classes, Instances

- **objects**
  - represent ‘things’ from the real world, or from some problem domain (example: “the red car down there in the car park”)

- **classes**
  - represent all objects of a kind (example: “car”)

- **instances**
  - Individual copies of an object.

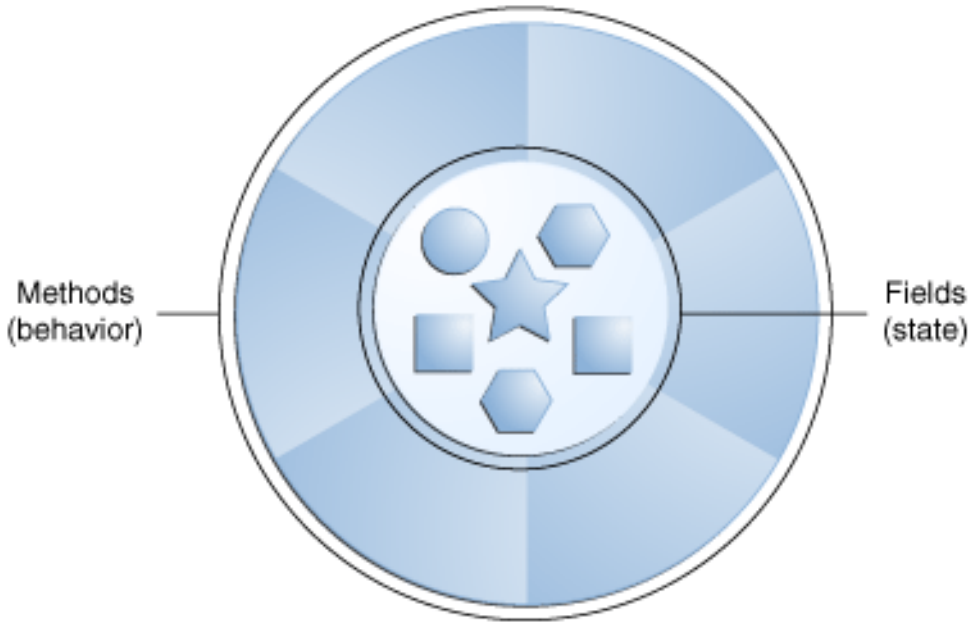
What is a Java Object?

- Objects have a state and behaviour.

State:
Name=x;
Age=y;
Colour=brown, yellow…;
hasBone=true/false;

Behaviour:
barking(postman),
fetch(bone),
wagging(tail),
…

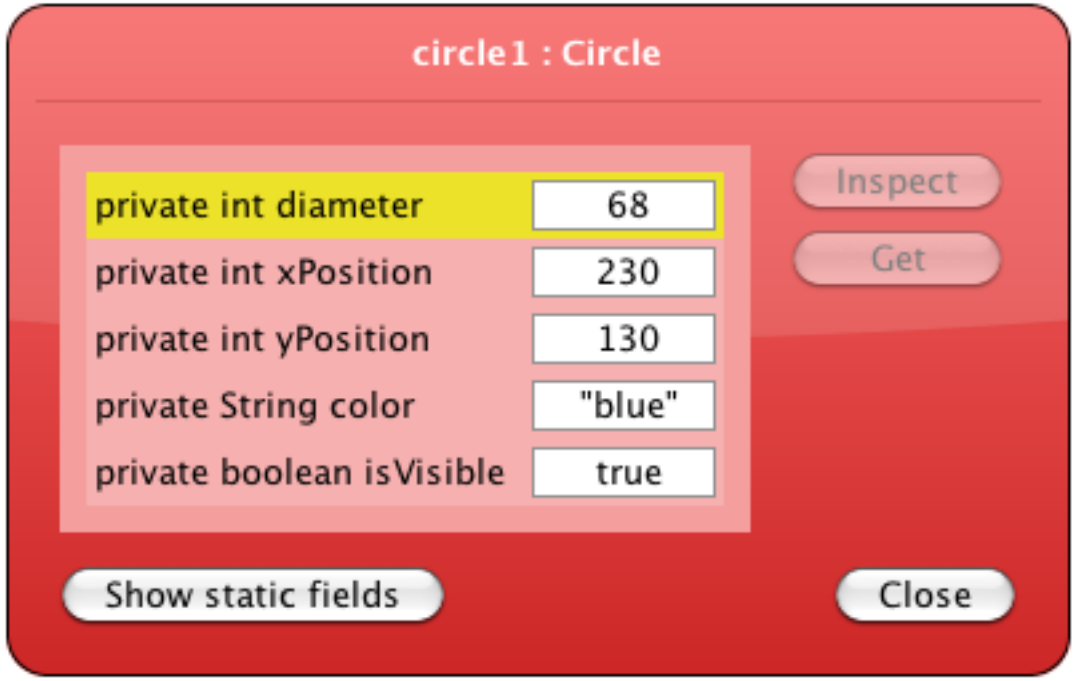
Real-world objects: multiple instances of “dog”.



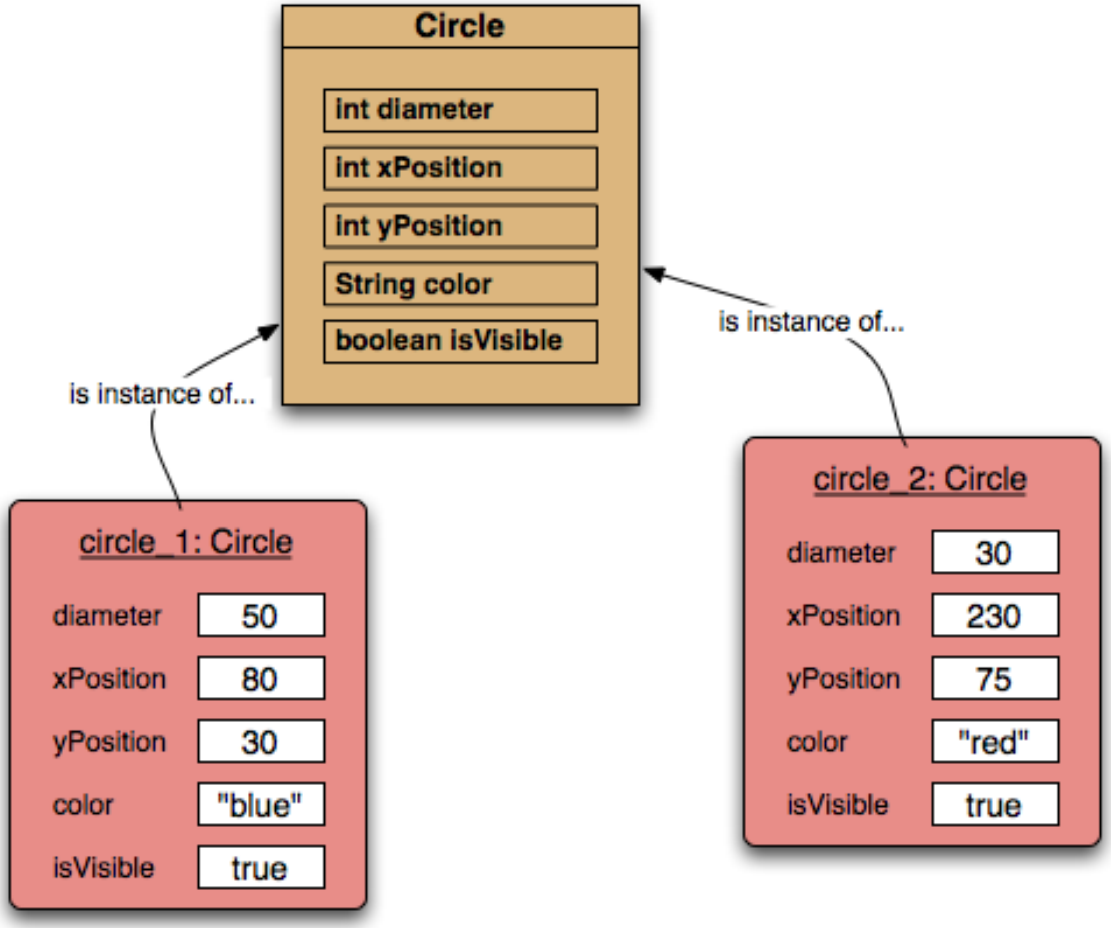
Software Object

http://docs.oracle.com/javase/tutorial/java/concepts/object.html

State of a “Circle”



Two circle objects



Summary: Class/Object/Instance?

- **Real world Objects** have a State and Behaviour.
- **Software Objects** have Fields and Methods.
- **A Class** is a "blueprint" to create objects.

```
public class Dog(){
//class body omitted   }
```

- **An Instance** is a unique copy of a class.

```
Dog fido = new Dog("Fido");
```

Other observations

- Many ***instances*** can be created from a single class.
- An object has ***attributes***: values stored in *fields*.
- The class defines what fields an object has, but each object stores its own set of values (the ***state*** of the object).

Understanding class definitions

Looking inside classes

Main concepts to be covered:

- fields
- constructors
- variable assignments

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

Declaring classes

```
public class MyClass {
    // fields
    //constructor
    // method declarations
}
```

- Class Names: nouns starting with upper case, e.g. “Dog”, “Person”, “Display”…

Example Implementation

```
public class Dog
{
   // Characteristics shared by all dogs (class variables).
   // The max age to which a dog can live.
   private static final int MAX_AGE = 18;

   // Individual characteristics (instance fields).
   // The dog's name.
   private String name;

   public Dog(String myName)
   {
      age = 0;
      name= myName;
   }

   private void fetch(Object o){
       […]
   }
}
```

DECLARATION

FIELDS

CONSTRUCTOR

METHODS

Keywords

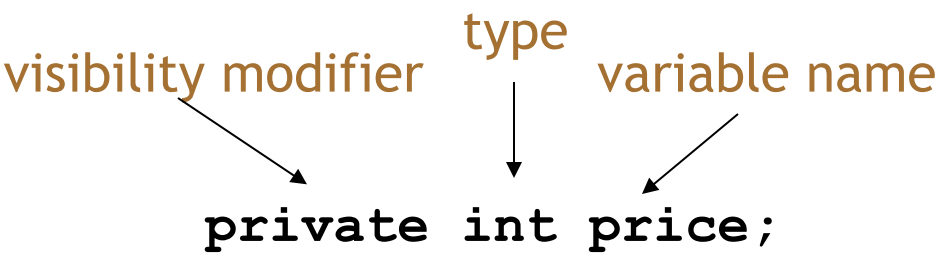
- Words with a special meaning in the language:
  - **`public`**
  - **`class`**
  - **`private`**
  - **`int`**
- Also known as *reserved words*.
- For a complete list see: http://docs.oracle.com/javase/tutorial/java/nutsandbolts/_keywords.html

Fields

- Fields store values for an object.
- They are also known as instance variables.
- Fields define the **state** of an object.
- Some values can change (instance fields).
- Some change not at all (class variables)

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

    Further details omitted.
}
```



Fields have Types

What is the Type of the following Fields?

- `Private int count;`
- `Private Student representative;`
- `Private Server host;`

What are the names of the following fields?

- `Private boolean alive;`
- `Private Person tutor;`
- `Private Game game;`

Visibility/ Access Modifiers

- public modifier:
  - the field is accessible from all classes.
- private modifier:
  - the field is accessible only within its own class.

- Fields are (usually) private.
- Constructors are public.

Constructors

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

- Initialize an object.
- Have the **same name** as their **class**.
- Close association with the fields.
- Store initial values into the fields.
- External parameter values for this.

Assignment

- Values are stored into fields (and other variables) via assignment statements:
  - *variable* = *expression*;
  - `price = 5;`
  - `price = cost;`
- A variable stores a single value, so any previous value is lost.

Assignments: What is the value?

```
int price = 500;
int total = 400;

total = price;
System.out.println("Total:" + total);
```

Total: 500

```
price = total;
System.out.println("Price:" + price);
```

Price: 500

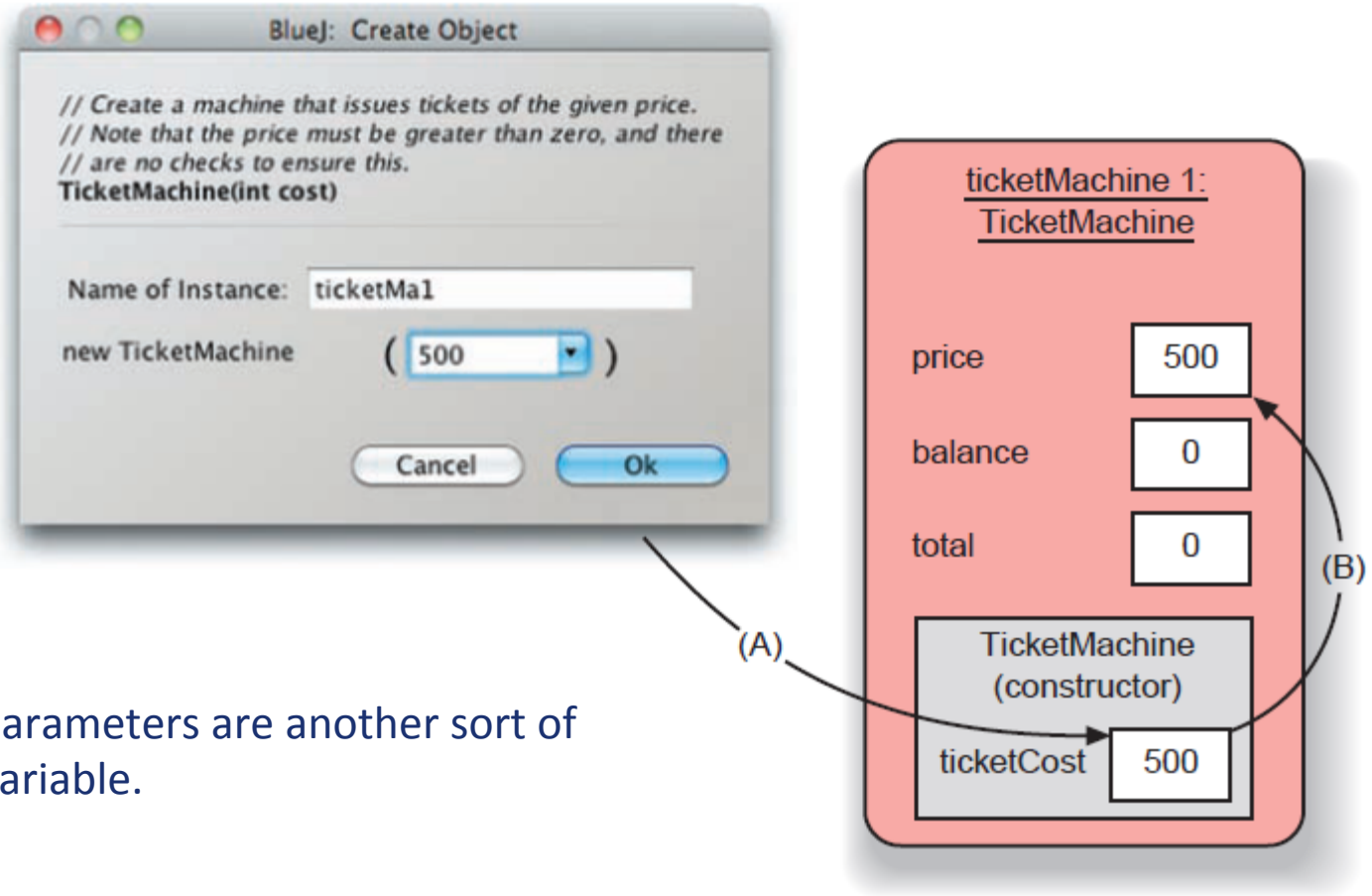
```
total +=price;
System.out.println("Total:" + total);
```

Total: 1000

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`**

Passing data via parameters



Parameters are another sort of variable.

Constructors with Parameters

Quiz:

Suppose the **class Pet** has a **field** called **name** of type **String**. Assign the field to the constructor's parameter.

```
public class Pet{
   // the name of the pet
   String name;
   // constructor for creating a new pet
   public Pet(String petName){
         name = petName;
   }
}
```

Creating new Objects

New Java runtime objects are created by:

1. Using the new operator
2. Calling the constructor of a class.
3. Assigning this new instance to a variable of the same type.

Object creation:

```
Dog fido = new Dog("Fido");
```

Associated constructor:

```
public Dog(String name);
```

Quiz: Creating new Objects

Quiz:

The following object creation will result in the ***Constructor*** of the **Date class** being called. Write the Constructor's header.

```
Date birthday = new Date(14,"March",1861);
```

```
public Date(int day, String month, int
year);
```

Summary

- **Java classes** create runtime **instances** which represent **real-world objects.**
- **Class bodies** contain fields, constructors and methods.
- **Fields** store values that determine an object's state.
- Fields are **variables** which store a single value.
- **Constructors** initialize objects – particularly their fields (e.g. via **parameters**).

Outlook

Understanding class definitions:

- methods
  - including accessor and mutator methods
- variable scope
- local variables

Homework

- Read chapters in “Objects First” (5th Edition):
  - 1 (all)
  - 2.1-2.6.