Introduction

OOP1 Review

COMP3607 Object Oriented Programming II

04-Sept-2019

Outline

- Course overview
- Review
 - OOP1 Concepts
 - Object-Oriented Design Principles

Review

- Classes vs Objects
- Instantiation, Initialisation
- Encapsulation and Abstraction
- Associations
- Inheritance
- Polymorphism

Classes vs Objects

A class defines a set of attributes and behaviour. An object is a concrete manifestation of a class.

Class	Object
1) Class is a collection of similar	1) Object is an instance of a
objects	class
 Class is conceptual (is a template) 	2) Object is real
 No memory is allocated for a	 Each object has its own
class.	memory
 Class can exist without any	 Objects can't exist without
objects	a class
5) Class does not have any	5) Every object has its own
values associated with the	values associated with the
fields	fields





Identify the classes

String s; int sum = 1000; ArrayList list = new ArrayList();



Classes vs Objects

Solution

```
String s;
int sum = 1000;
ArrayList list = new ArrayList();
```

IDEs can help us out with colour coded classes

5 5 5 * * X 5 7 9 9 8 * * *	1 @ experimentary 1 > 8 @ 2 1 0 17 11 10 10 10 10 10 10 10 10 10 10 10 10
Gereiner auf Dan Dinen Daues Born B	
	👷 Bandhillyania = 🕲 Taerdenizania = 🔒 Bradhillyania = 😢 Bradhillyania = 😢 Bradhillyania = 😢 Bradhillyania = 😢 Bradhillyania
· Evenjevrapi	1 pedage castwojew.opt
* Dan	
 Boginespherenumber 	a disport and
• Designer	
Per la	12 (ubject the jacobertor to (
Automator	38 0 aldert Reedederberbert ertent schedulberberbert
Automas	25 d Sailleft skjatt Pasedaskerfarnat antalds Asst2assfarmet(Pasedasker) (26 d artis(skjat Pasedasker)s 2x0xlas = 2x0xfag(skj.dermet(National))
Aptendations and	20 ** def writigidge Menefanker)e Juitales * Juiferingidgidermet(National)) 20
Init	24 det matigines biblioje Bandader # 222
Crospilente	
Coupdevioles.cols	
Organizationslervice	28 0 Suitfeit aldett Pheedhakersformet antends foor2oorderent[imgPheedhaker]] (
 Qastap.sola 	14 def write(dd): tes(Meenholer)): 2-Weine it seefweet(Meenholer).write(dd))
Ensimplervice Densimplervice	
• regardeenalees	and read(see history) + 22
Tegarlaman	
G forjaufora/ornation	
• Deniv	31 0 skjert Plandericelystack artests befaulthantystack (
• Dainan	23 Separa Provider Conditionality
Contraction of the second second	34 0 implicit abdect Phanabericalarmat entands hasthianformat(Phanaberics) (
Classifierity Classifierity	an et a de artistate maneterio a amber a antiertime
E trailed was agin	M Not a signal and
Transfordations	27 "structurtion" sk obj.constation.ang(scher str Admater(scher.getting())).getterUse(hubul), 28 "structurtion" sk obj.constructure.ang(scher str Admater(scher.getting())).getterUse(hubul),
 Cross sols 	Image: Sector Control (Sector (
Crapterinfered	Contraction of the International Contraction
Cathoritety	"evertiable" = . Advelance(of evertiable)
 Comparisation acuts 	
 GranizationExtension.scala 	
 OrganizationExtensioned.ecals 	stid def realizione brinkeje Resolution a III
 Speckage stats ProveDevice stats 	
Provehumber scale	
- Inglish	40 0 Salleli skjet Panderlesfanst atals fortisedenst(https://wederlesj) (40 / of artistalj fort/headerlesj); Salle z attisent/headerlesj);
Vingiliain V Carasteracaia	
and the second s	and def margines include intermedical a 202
O Section	
 Coher.sola 	
 Eperaterica 	
 El security El security 	Bi () algest includent/vessel astanda infactions/vessel (
 Contage scale 	55 Sept Finder Laboration
Baranhar	16 0 Sutterin abjest berforfunst methods Austichenst(beries) (17.4 0 def writedelde Berlande Austein 6 old autein (
C application.com	In the device president of the device states
Calept properties	
 Libraries 	
\$1 1000 \$1 Charges	



Exercise

Identify the objects and classes

```
String s;
double delta = 1.89;
boolean equal = false;
Object object;
Integer count = new Integer(300);
```



Exercise

Identify the objects and classes

```
String s;
double delta = 1.89;
boolean equal = false;
Object object;
Integer count = new Integer(300);
```

Declaration

The process of associating an object variable name with an object type (class).

String s; ArrayList list; Scanner s; StringTokenizer st;

Instantiation

The process of creating a new instance of a class. This is done in order to use the services of a class. The **new** keyword is used.

```
String s = new String();
ArrayList list = new ArrayList();
Scanner s = new Scanner();
StringTokenizer st = new StringTokenizer();
```

Initialisation

The process of assigning state (values) to an instance of a class.

String s = new String("Clear Tape");
ArrayList list = new ArrayList(10);
Scanner s = new Scanner("data.txt");
StringTokenizer st = new StringTokenizer(";");



Exercise

Distinguish between instantiation, declaration and initialisation

```
String s;
s = "Clear Tape";
ArrayList stationery;
stationery = new ArrayList();
new File("stocks.txt");
String message = new String("Out of Stock");
```



Exercise

Solution

Distinguish between instantiation, declaration and initialisation

String s; // Declaration
s = "Clear Tape"; // Initialisation
ArrayList stationery; // Declaration
stationery = new ArrayList(); //Instantiation
new File("stocks.txt"); //Instantiation, no Declaration!
String message = new String("Out of Stock"); //all 3

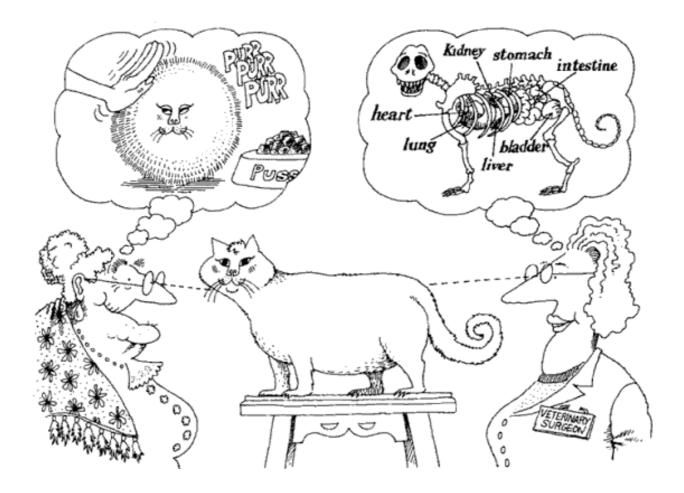
Abstraction

"Abstraction is one of the fundamental ways that we as humans cope with complexity." (Booch, 1994).

- Recognition of similarities
- Emphasis on significant details
- Independent of implementing mechanism

Abstraction

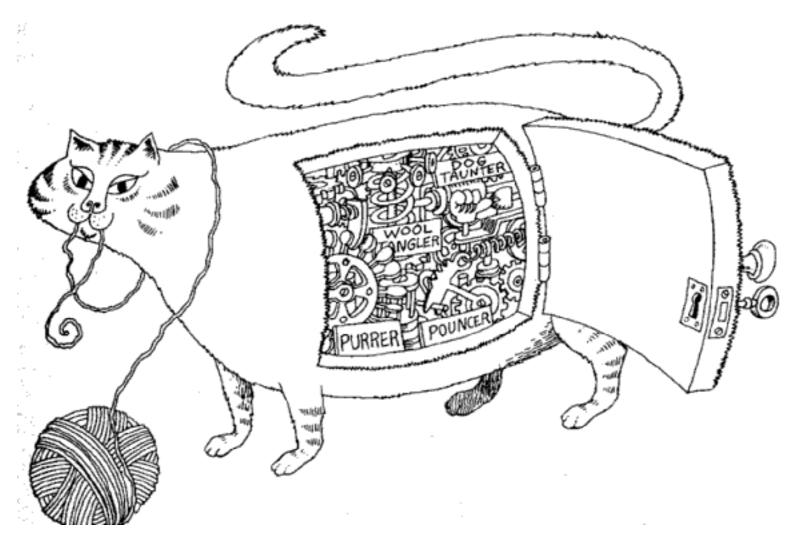
An abstraction denotes the **essential** characteristics of an object that **distinguish** it from all other kinds of objects and thus provide crisply defined **conceptual boundaries**, relative to the perspective of the viewer.



Observable behaviour

Encapsulation

Encapsulation focuses upon the implementation that gives rise to observable behaviours.



Hides the details (implementation) that gives rise to observable behaviour

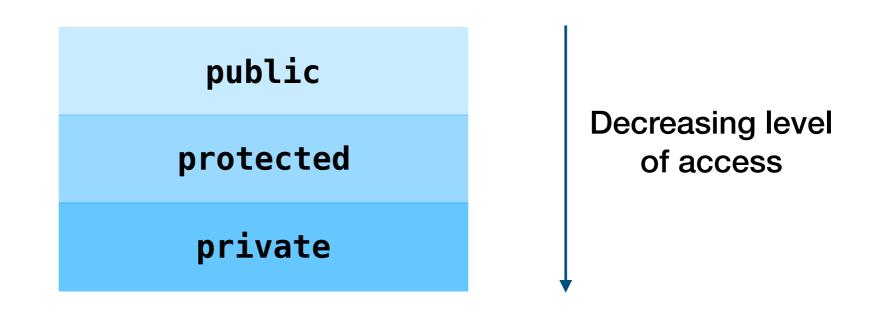
Information Hiding

Encapsulation is most often achieved through information hiding.

Information Hiding is the process of hiding all the secrets of an object that do not contribute to its essential characteristics:

- the structure of an object is hidden
- the implementation of its methods is hidden

Access Modifiers



- Public: A declaration that is accessible to all clients
- Protected: A declaration that is accessible only to the class itself, its subclasses, and its friends
- Private: A declaration that is accessible only to the class itself and its friends

Interface vs Implementation

The interface of a class provides its outside view and therefore emphasizes the abstraction while hiding its structure and the secrets of its behavior.

This interface primarily consists of the declarations of all the operations applicable to instances of this class, but it may also include the declaration of other classes, constants, variables, and exceptions as needed to complete the abstraction.

Interface vs Implementation

By contrast, the implementation of a class is its inside view, which encompasses the secrets of its behaviour.

The implementation of a class primarily consists of the implementation of all of the operations defined in the interface of the class.

Relationships

Classes, like objects, do not exist in isolation. Very often, an object-oriented program consists of a set of interacting objects whose classes are related in some way.

Relationships between classes are established to either:

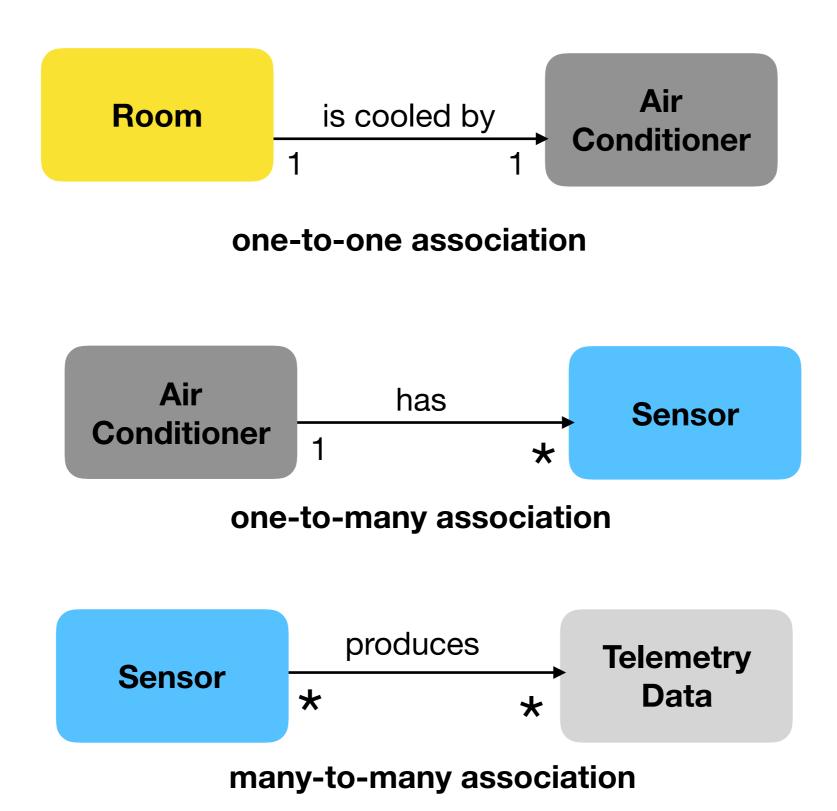
- Indicate some sort of sharing between the classes
- Indicate a semantic connection between the classes

Kinds of Relationships

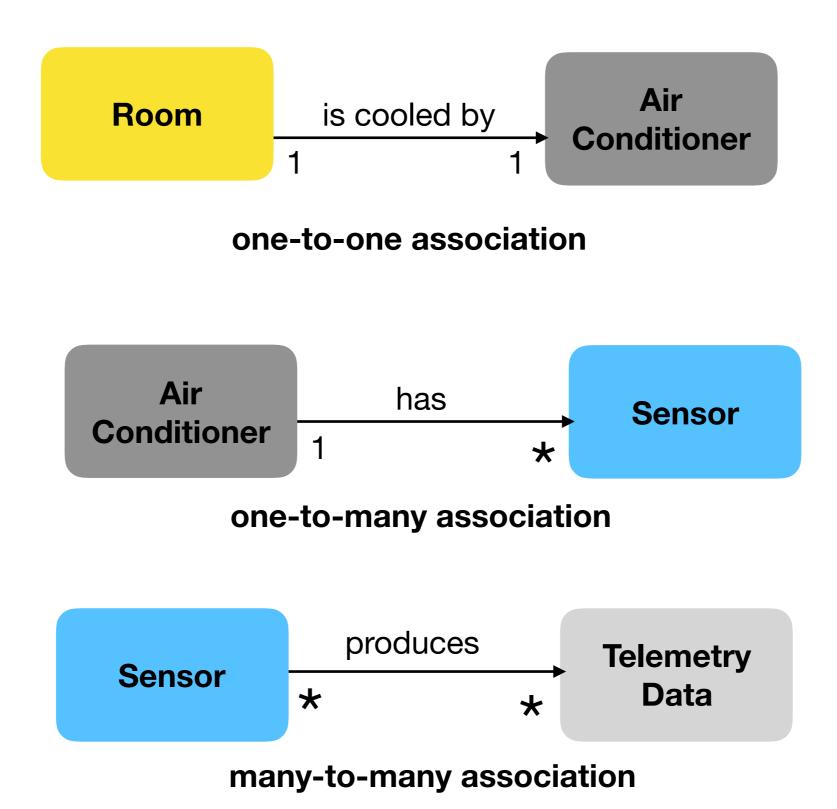
There are three basic kinds of relationships:

- 1. Association/Dependency (uses)
- 2. Generalisation/Specialisation (is-a)
- 3. Composition/Aggregation (part-of)

Example-Association

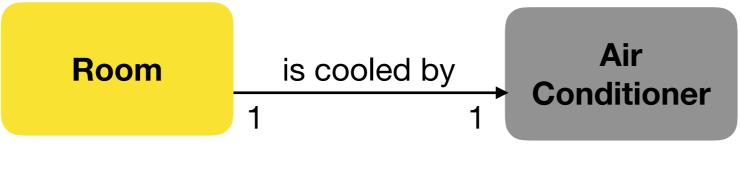


Example-Association



Example - Implementing Associations

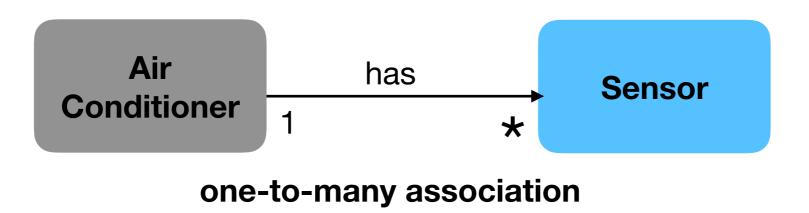
```
public class Room{
    private AirConditioner ac;
    public Room( ){
        ac = new AirConditioner( );
    }
}
```



one-to-one association

Example - Implementing Associations

```
public class AirConditioner{
    private Sensor[] sensors;
    public AirConditioner( ){
        sensors = new Sensor[3];
    }
}
```





Exercise

Identify the objects and their associations in the following scenario.

A point-of-sale (POS) system is a computerised application used in a retail store to record sales and handle payments.





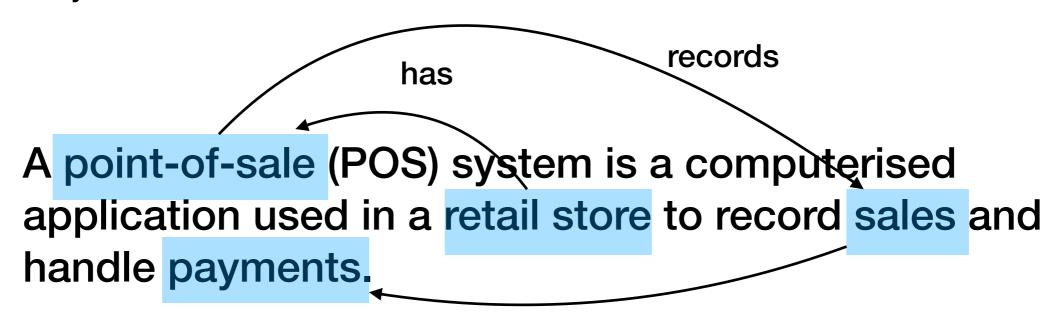
Identify the objects first

A point-of-sale (POS) system is a computerised Objects application used in a retail store to record sales and handle payments.



Exercise

Next, identify the main associations between the objects.



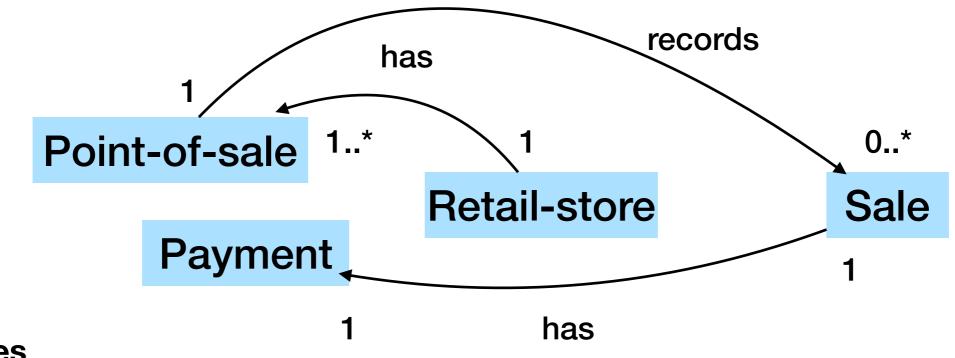
has

Associations





Clean up and assign cardinalities.



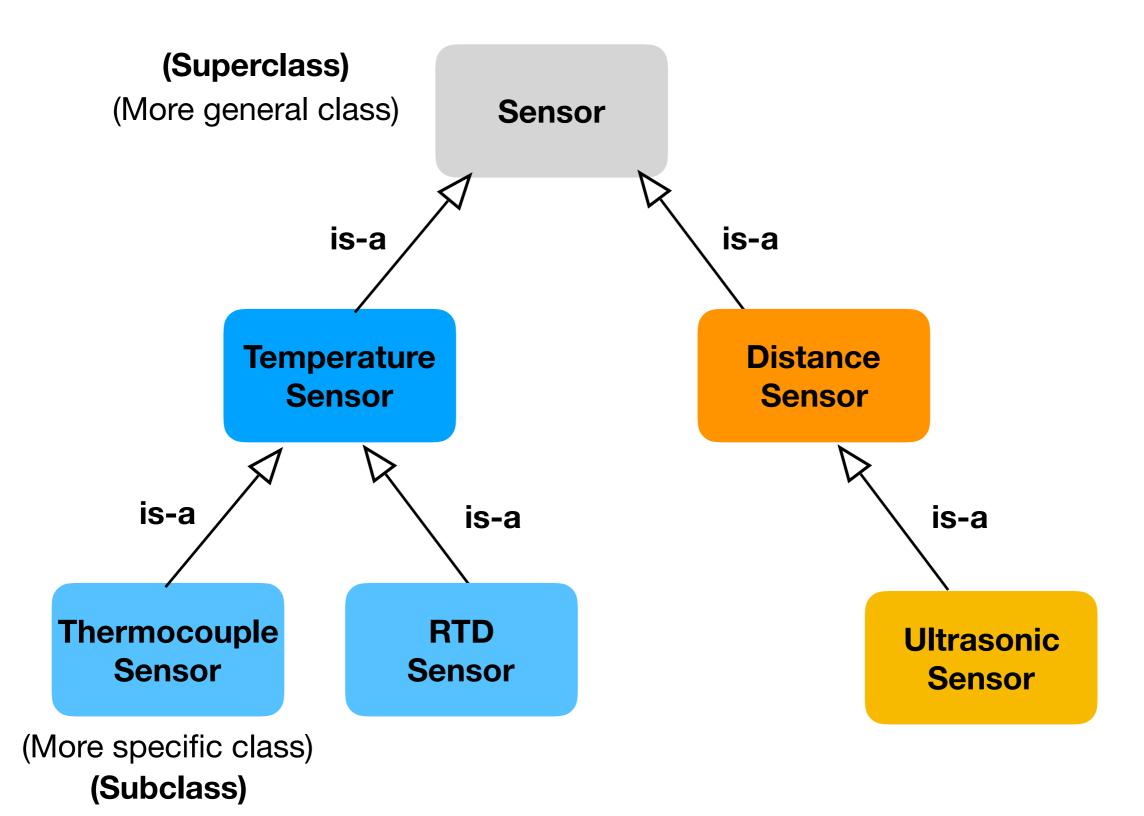
Cardinalities

Inheritance

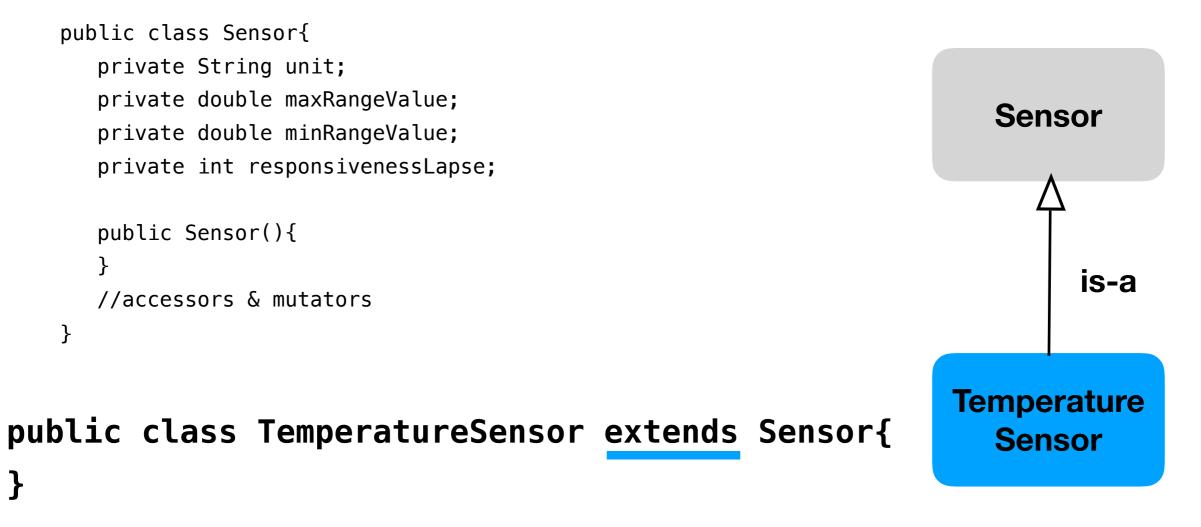
Inheritance is a relationship among classes wherein one class shares the **structure** and/or **behaviour** defined in one (single inheritance) or more (multiple inheritance) other classes (Booch 1994).

A class from which another class inherits its structure and/or behaviour is called the **superclass**. A class that inherits from one or more classes is called a **subclass**

Example - Inheritance

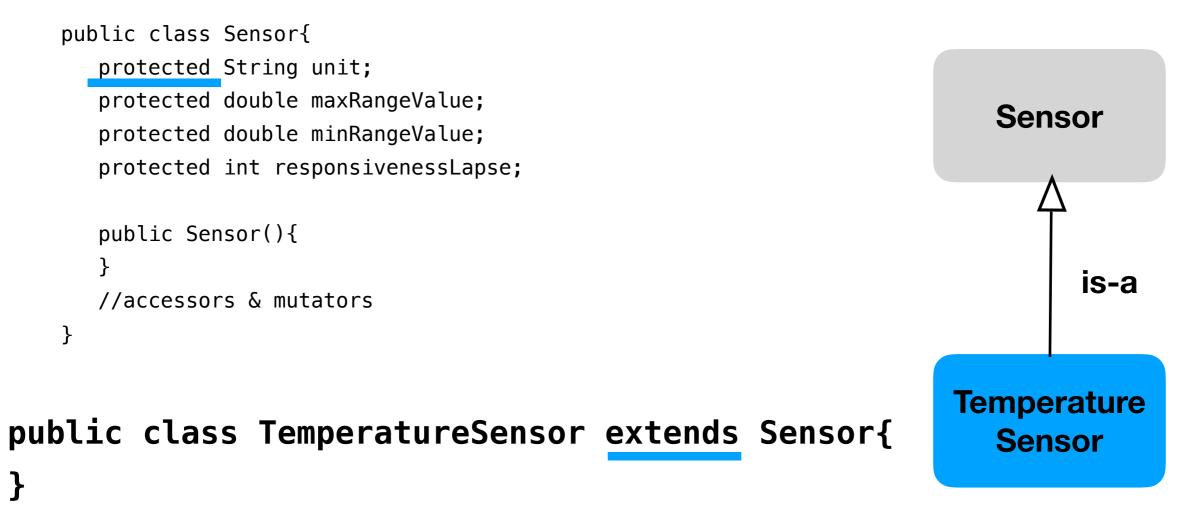


Example Implementing Inheritance



Client class

Example Implementing Inheritance



Client class



Exercise

Identify the objects and their relationships in the following scenario.

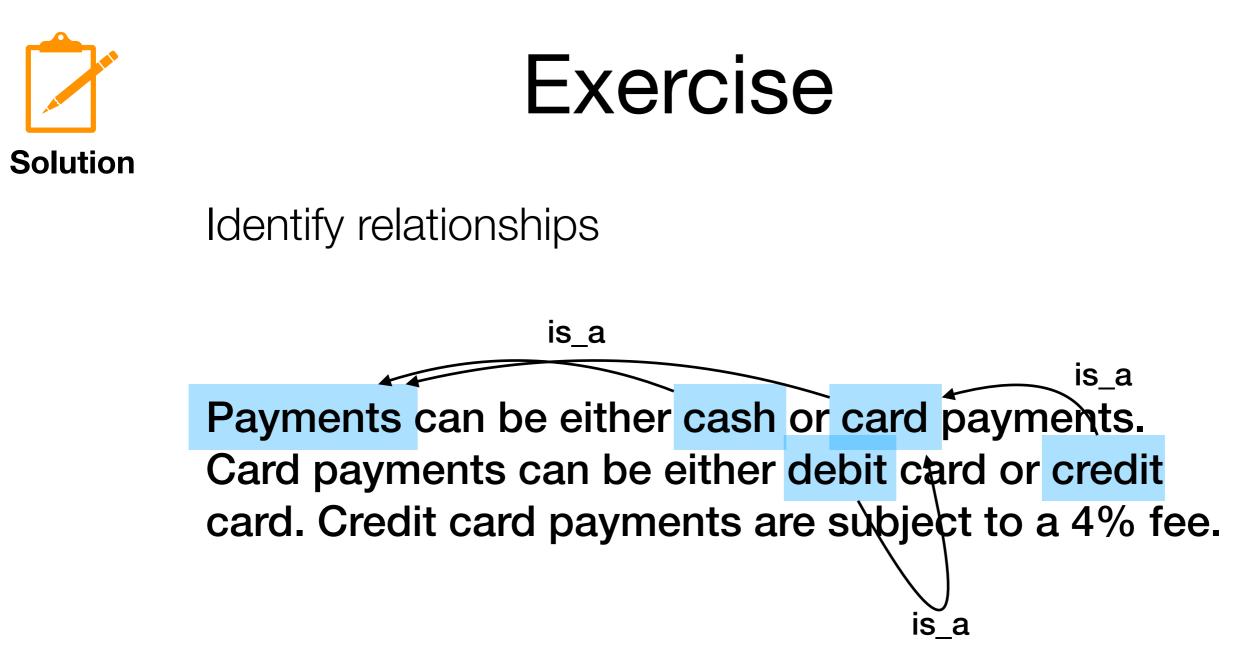
Payments can be either cash or card payments. Card payments can be either debit card or credit card. Credit card payments are subject to a 4% fee.



Exercise

Identify the objects

Payments can be either cash or card payments.Card payments can be either debit card or creditObjectsCard. Credit card payments are subject to a 4% fee.

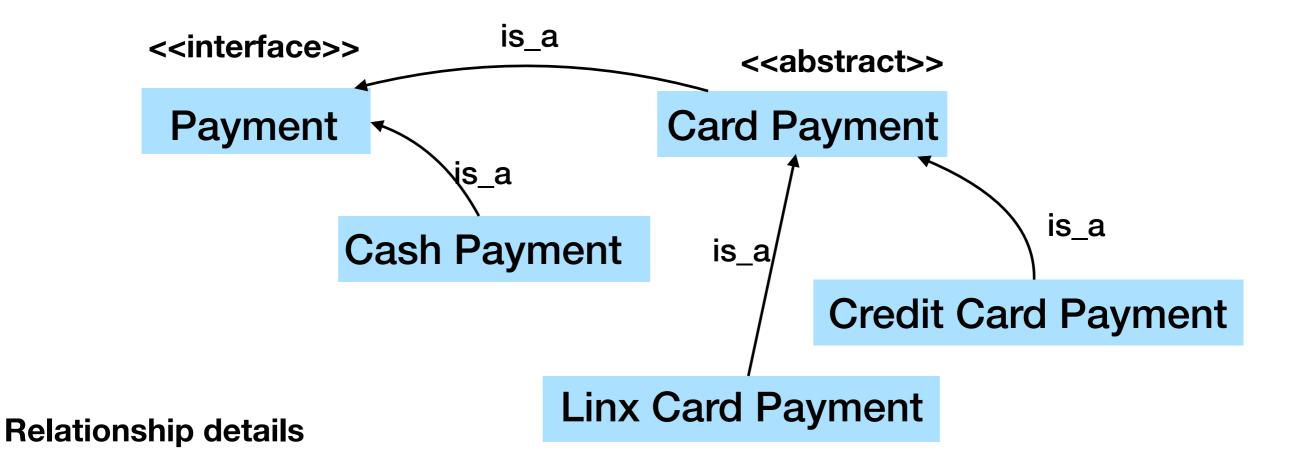


Relationships





Fine tune



Polymorphic Objects

When the static type and the dynamic type of an object are different, that object is said to be **polymorphic**.

A polymorphic object exhibits different behaviour based on the differences between its static type and dynamic type.

Example - Polymorphic Objects

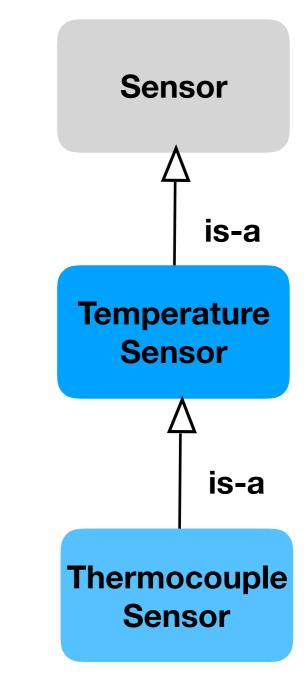
Sensor ts = new ThermocoupleSensor();
//ST: Sensor, DT: ThermocoupleSensor

TemperatureSensor ts = new TemperatureSensor();

//ST: TemperatureSensor, DT: TemperatureSensor

Object o = new ThermocoupleSensor();
//ST: Object, DT: ThermocoupleSensor

(Only 2 objects are Polymorphic)



Abbreviations:

ST - Static type; DT - Dynamic type

References

- Booch, Grady. (1988) OBJECT-ORIENTED ANALYSIS
 AND DESIGN
- Mohan, Permanand (2013) FUNDAMENTALS OF
 OBJECT-ORIENTED PROGRAMMING IN JAVA