Fundamental Programming Help session 3 – UML

Federica Comuni
federica.comuni0002@stud.hkr.se

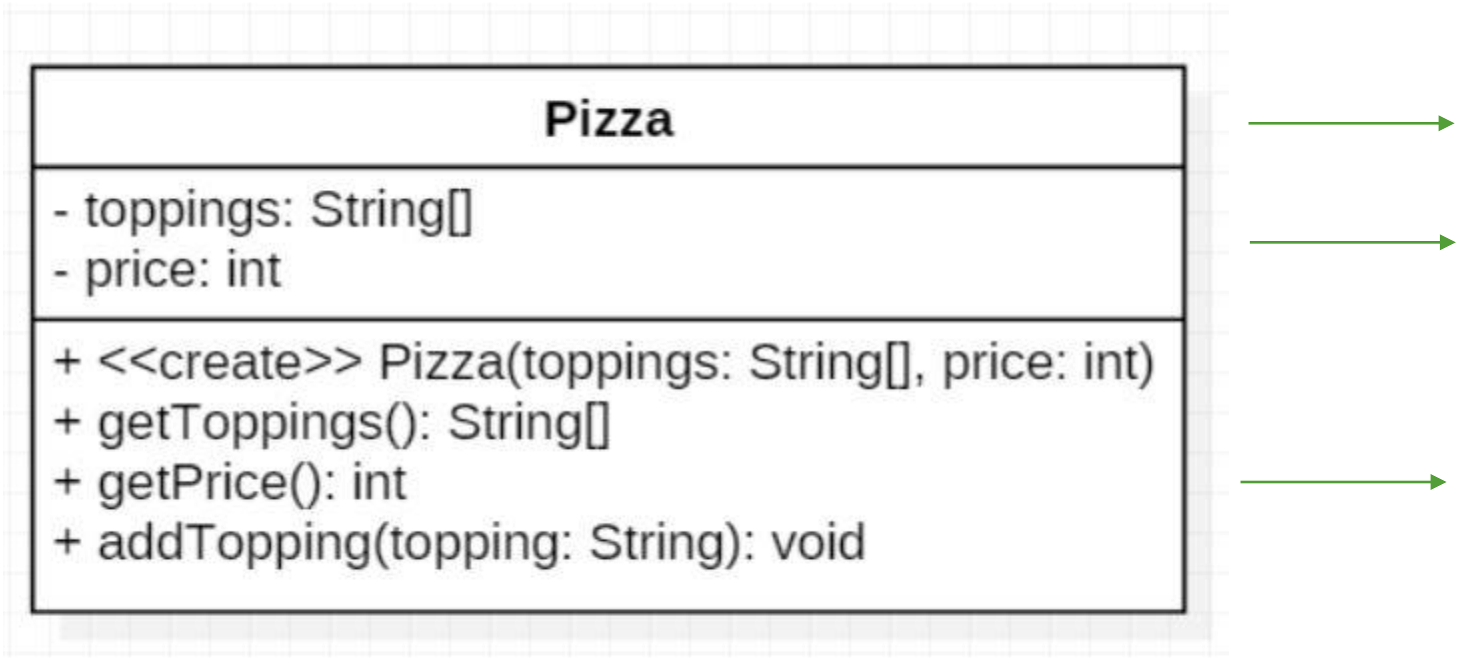
What is UML?

UML (Unified Modeling Language) is a modeling language that specifies a set of **diagrams** that help developers **visualizing** and **documenting** software systems

It helps us understand which **classes** a program has, what is the **relationship** between the classes, and what are the **fields** and **methods** of each class

Högskolan Kristianstad

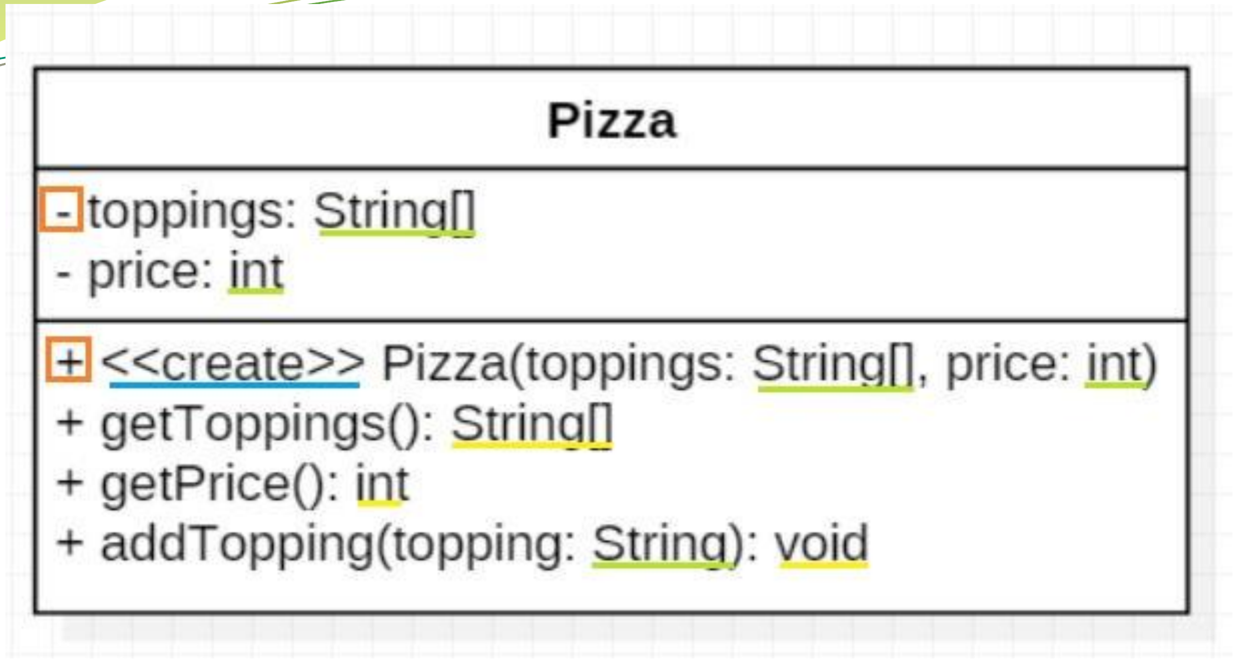
A class is represented by a **rectangle** with three sectors:



Pizza

- toppings: String[]
- price: int

+ <<create>> Pizza(toppings: String[], price: int)
+ getToppings(): String[]
+ getPrice(): int
+ addTopping(topping: String): void



```
Pizza
-toppings: String[]
- price: int
+<<create>> Pizza(toppings: String[], price: int)
+ getToppings(): String[]
+ getPrice(): int
+ addTopping(topping: String): void
```

Orange: **access modifiers**
(-) minus = private (for fields)
(+) plus = public (for methods)

Green: **datatype** for fields and methods' parameters

Yellow: **return type** for methods

Blue: **constructor tag**

private String[] toppings;

private int price;

public Pizza(String[] toppings, int price) {…}

public String[] getToppings() {…}

public int getPrice() {…}

Three important things to remember

- Variable name : datatype

- The **return type** of the method (**nothing in the constructor, void if the method is void**)

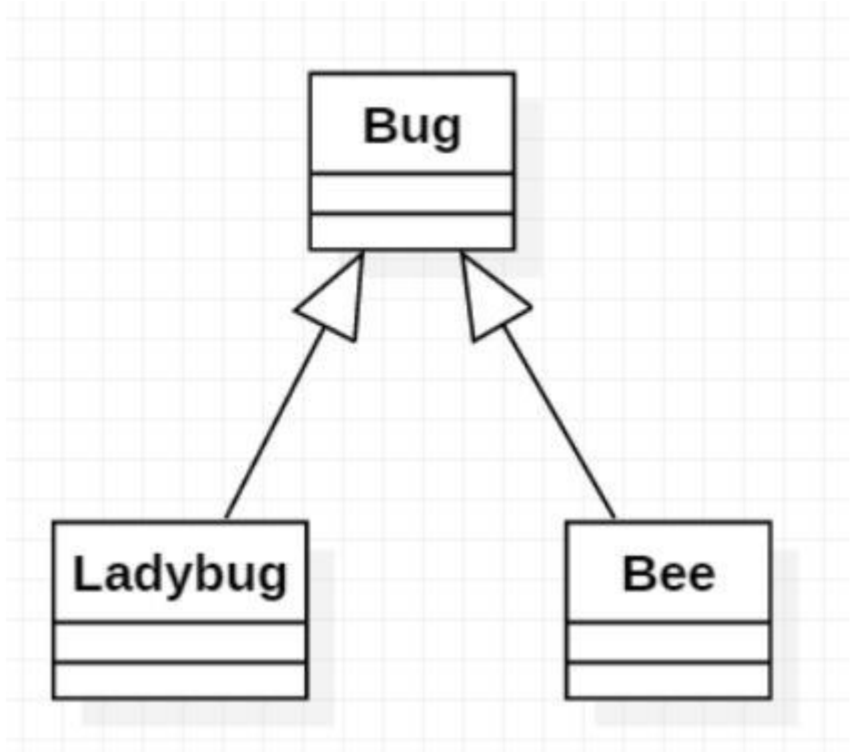
- The **constructor** (with the **<<create>>** tag)

Högskolan
Kristianstad

Relationships between classes

Inheritance – "**is a**" relationship

```
public class Bug {…}
public class Ladybug extends Bug {…}
public class Bee extends Bug {…}
```

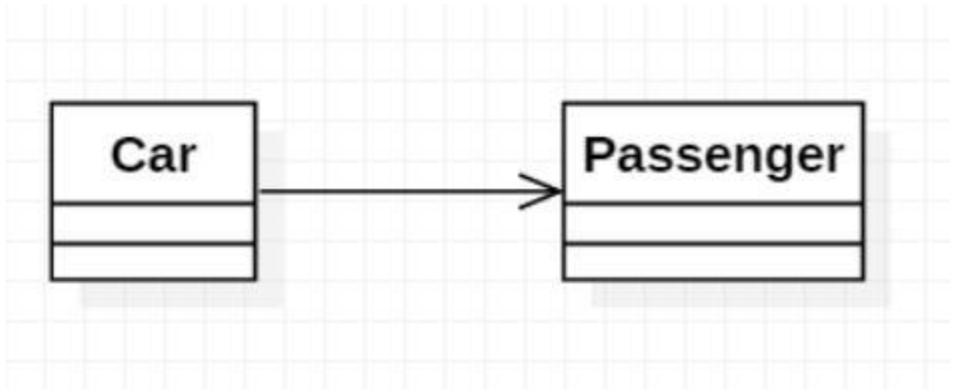


Högskolan Kristianstad

Relationships between classes

Direct association – “**has a**” relationship

```
public class Car {
    private Passenger passenger;
}
```

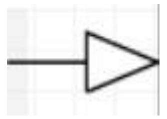
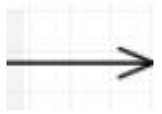


Högskolan Kristianstad

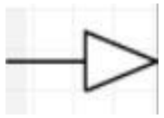
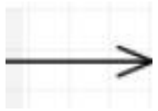
The hardest part

Translating a given text into UML class diagram – how do we know what are the classes, the fields, the methods, and the relationships between classes?

Högskolan Kristianstad

We can think of classes as **objects** of the real life	**Pizza**
Fields are their **characteristics**, or something that the object **has** (even another object)	**price, toppings**
Methods are **actions** that the object does, or **actions** that other objects can do with it	**addTopping()**
If a specific object **is** an individual of another object -> the relationship between the 2 objects is **inheritance**	**Pizza**  **Food**
If an object **has** another object as instance variable -> the relationship between the 2 objects is **association**	**Pizza**  **Topping**

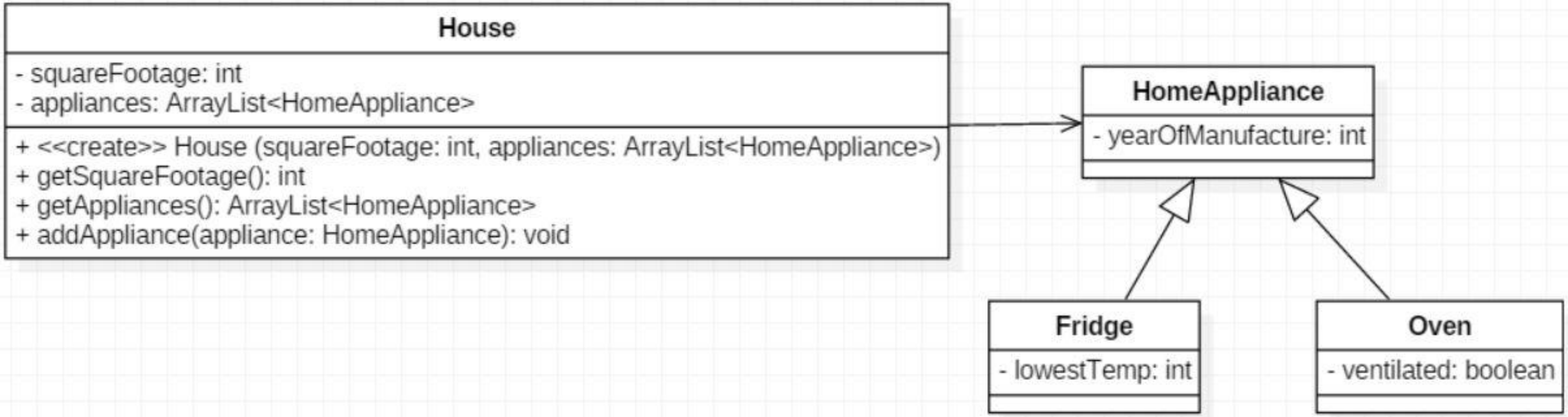
To remember:

- Arrow with closed point for inheritance 
- Arrow with open point for association 

A program describes a house. A house has a square footage and some home appliances (stored in an ArrayList). All home appliances store the year in which they were manufactured. There are two types of home appliances: a fridge and an oven. The fridge stores information about the lowest temperature it can reach, while the oven stores if it is ventilated or not.
In the house you can add new home appliances by specifying which home appliance you want to add.
The square footage and appliances of the house are passed through the constructor, and cannot be modified once an instance of House has been created. They can however be accessed through getter methods.

Draw a UML class diagram for this program and write Java code for it.

Högskolan Kristianstad



House
- squareFootage: int
- appliances: ArrayList<HomeAppliance>
+ <<create>> House (squareFootage: int, appliances: ArrayList<HomeAppliance>)
+ getSquareFootage(): int
+ getAppliances(): ArrayList<HomeAppliance>
+ addAppliance(appliance: HomeAppliance): void
HomeAppliance
- yearOfManufacture: int
Fridge
- lowestTemp: int
Oven
- ventilated: boolean

Högskolan Kristianstad

```java
import java.util.ArrayList;

public class House {

  private int squareFootage;
  private ArrayList<HomeAppliance> appliances;

  public House(int squarefootage, ArrayList<HomeAppliance> appliances) {
   this.squareFootage = squareFootage;
   this.appliances = appliances;
  }

  public int getSquareFootage() {
   return squareFootage;
  }

  public ArrayList<HomeAppliance> getAppliances() {
   return appliances;
  }

  public void addAppliance(HomeAppliance appliance) {
   appliances.add(appliance);
  }
}
```

```java
public class HomeAppliance {

  private int yearOfManufacture;

}
```

```java
public class Fridge extends HomeAppliance {

  private int lowestTemp;
}
```

```java
public class Oven extends HomeAppliance {

  private boolean ventilated;
}
```