

CSSE2003

Software Engineering Studio

Lecture 6 UML Class Diagrams

Supplementary Materials Sample Solutions

Semester 2, 2009

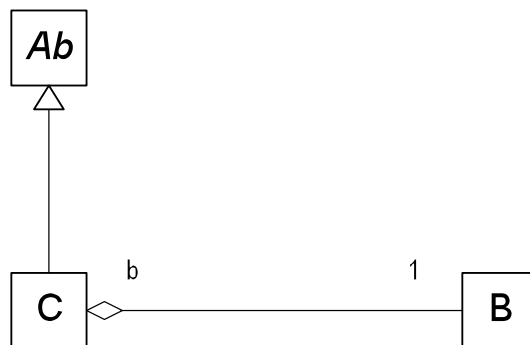
School of Information Technology and Electrical Engineering
The University of Queensland

Exercise 1

Draw a class model for these classes. [Braude, Ex. 3.2, p80]

```
abstract class Ab {}
class B {B() {}}
class C extends Ab {B b = new B();}
```

Sample Solution



Exercise 2

Draw a class model for these *AutoApplication* classes. You don't need to include standard system input and output classes or the HashTable class (but you could).

```
class AutoApplication1 {
private static Automobile auto;
private static void getAutoTypeFromUser() throws IOException {
    Automobiles autoTable = new Automobiles();
    autoTable.put("ford", new Ford() );
    autoTable.put("chevy", new Chevy() );
    System.out.println( "Pick from the following:" );
    Enumeration en = autoTable.keys();
    while (en.hasMoreElements()) {
        System.out.println(en.nextElement());}
    BufferedReader bufReader =
    new BufferedReader( new InputStreamReader(System.in) );
    String autoType = bufReader.readLine();
    auto = (Automobile)autoTable.get(autoType);
    if(auto == null) {
        System.out.println("A ford is assumed.");
        auto = new Ford();
    }
}}
```

```

public static void main( String[] args ) throws IOException {
    getAutoTypeFromUser();
    auto.outputDescription();
}

abstract class Automobile {
    public abstract void outputDescription();
}

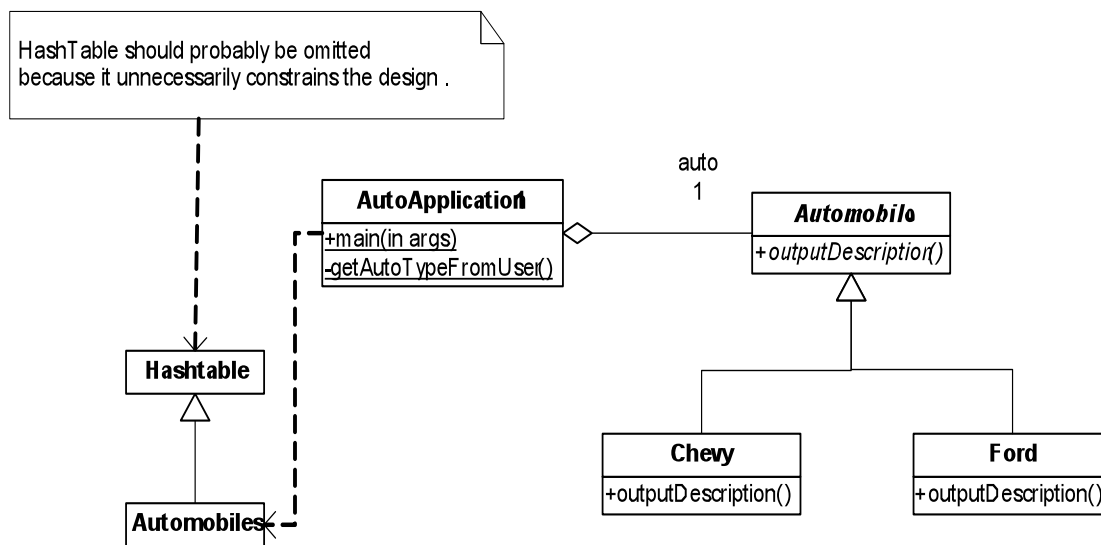
class Automobiles extends Hashtable {}

class Chevy extends Automobile {
    void outputDescription() {...}
}

class Ford extends Automobile {
    void outputDescription(){...}
}

```

Sample Solution



Notes:

1. The method `main` is underlined since it is static, and the type `(String[])` for the parameter `args` could be (but need not be) included.
2. The method `outputDescription` in class `Automobile` is in italics since it is abstract.

Exercise 3

Repeat the previous exercise using this class

```

class AutoApplication2 {

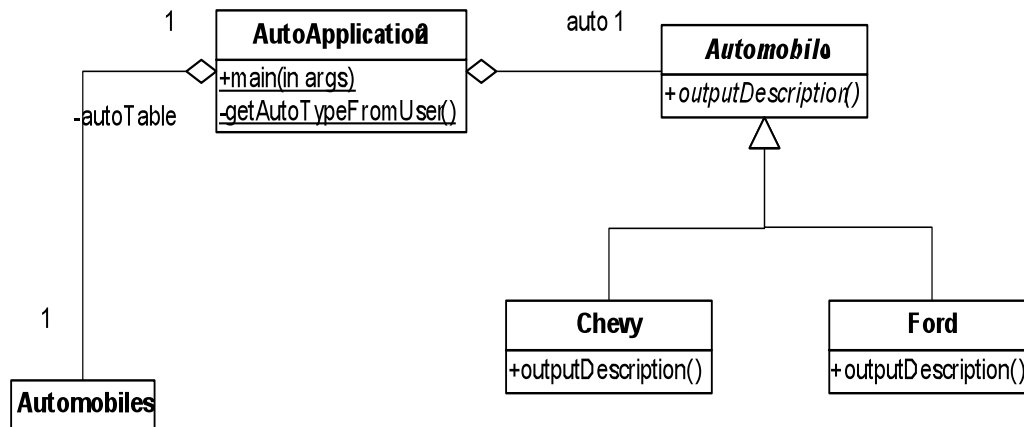
    private static Automobile auto;
    private static Automobiles autoTable;

    private static void getAutoTypeFromUser() throws IOException {
        autoTable = new Automobiles();
        ... as in Exercise 2 ...
    }

    public static void main( String[] args ) throws IOException {
        ... as in Exercise 2 ...
    }
}

```

Sample Solution

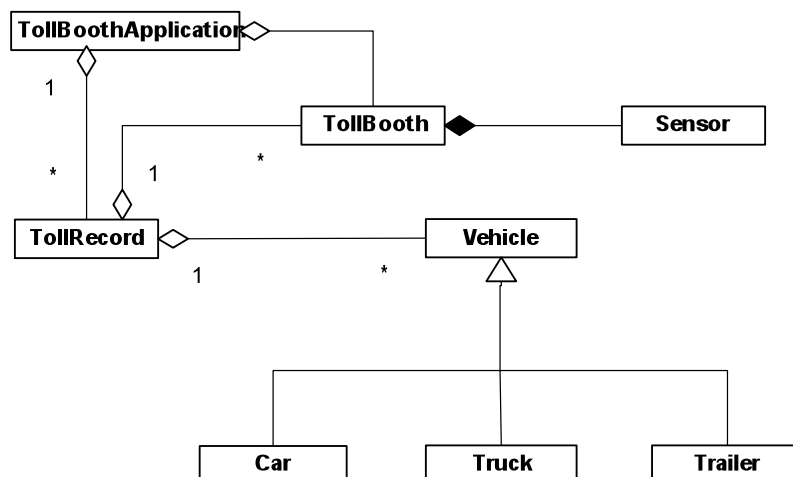


Notes:

1. The method `Main` is underlined since it is static, and the type (`String[]`) for the parameter `args` could be (but need not be) included.
2. The method `outputDescription` in class `Automobile` is in italics since it is abstract.

Exercise 4

Consider an automated toll application for collecting a toll at an unmanned booth. The minimum requirement is to store the identity of an automobile when it passes through the booth. Given the following classes for the application, *TollBoothApplication*, *TollBooth*, *Sensor*, *Vehicle*, *Car*, *Truck*, *Trailer*, draw a simple class diagram that captures plausible relationships among these classes. You may add extra classes. [Braude, Ex. 5.2.2, p121]



The class `TollRecord` has been introduced to record “a vehicle passing through a toll booth”. It is an example of a mediator (pattern).