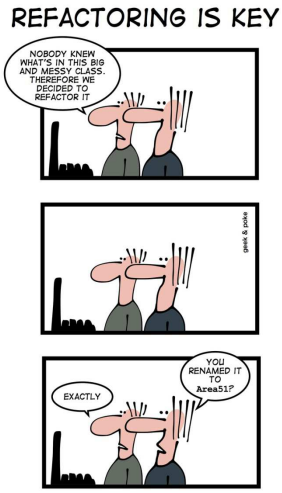


Data

www.cs.uoi.gr/~zarras/soft-devII.htm

Sources: M. Fowler Refactorings Catalog
W. Wake Refactoring Workbook

Primitive Obsession

Primitive Obsession

Symptoms

Look for:
in general, the problem is if we use variables/fields of primitive type instead of objects

uses of the primitive or near-primitive types (int, float, String, etc.)
constants and enumerations representing small integers
string constants representing field names

Causes (what kind of primitives can we have??)
Missing class: Since almost all data must be in a primitive somewhere, it's easy to start with a primitive, and miss an opportunity to introduce a new object.

Simulated field accessors: Sometimes, a primitive is ***used as an index****, to provide* access to pseudo-fields in an **array**. Strings are occasionally used this way with HashTables or Maps.

Type codes: instead of having different sub-classes of objects we have a class field whose values ***distinguish different kinds of objects****. That belong to the class*

Primitive Obsession

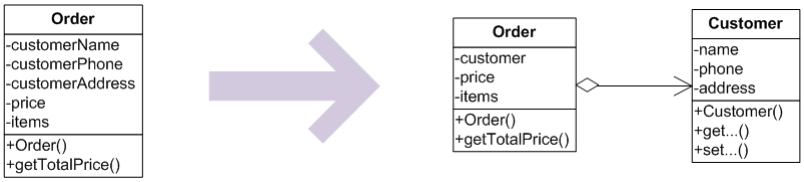
What to Do

For missing classes:
Replace Data Value with Object *to make data values "first-class."*

Replace Data Value with Object

You have a data item that needs additional data or behavior.

Turn the data item into an object.



Primitive Obsession

What to Do

For simulated field accessors:

If the primitives are indexes used to treat certain array elements as pseudo fields, ***Replace Array with Object***.

Replace Array with Object

You have an array in which certain elements mean different things.

Replace the array with an object that has a field for each element.

```
String[] row = new String[3];
row [0] = "Liverpool";
row [1] = "15";
```

The primitives here are the array and the indexes

```
Performance row = new Performance();
row.setName("Liverpool");
row.setWins("15");
```

Primitive Obsession

What to Do

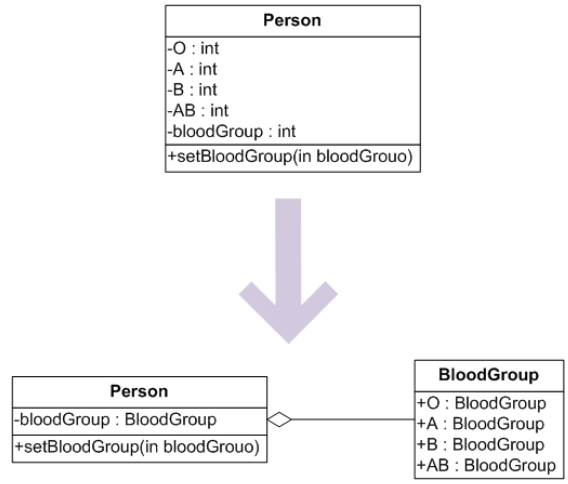
For type codes:.

If no behavior is conditional on the type code, then it is more like an enumeration, so ***Replace Type Code with Class***.

Replace Type Code with Class

A class has a numeric type code that does not affect its behavior.

Replace the number with a new class.



Primitive Obsession

What to Do

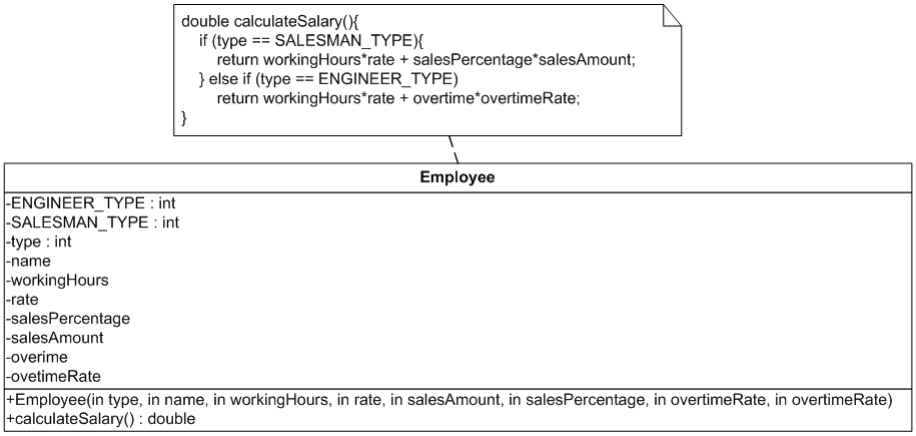
For simulated types: an integer type code stands in for a class.

If the **behavior of the object depends on the typecode** and the typecode value is **immutable during the lifetime of the object**, and the class is not subclassed already, ***Replace Type Code with Subclass***.

Replace Type Code with Subclass

You have an immutable type code that affects the behavior of a class.

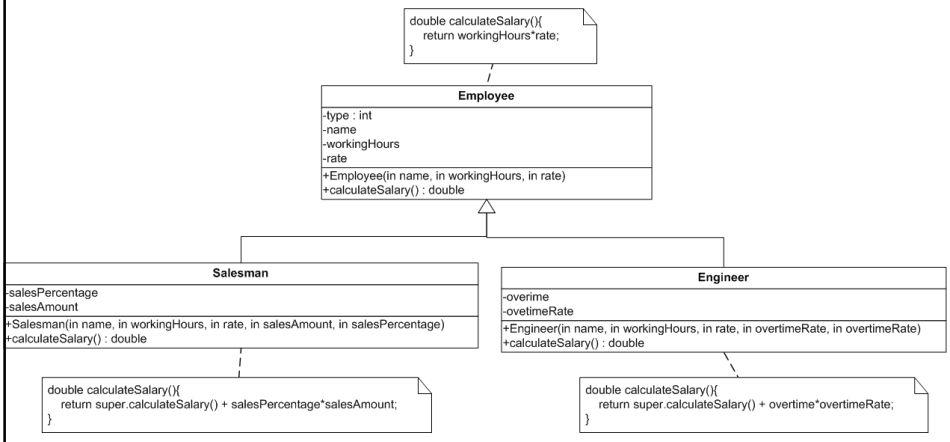
Replace the type code with subclasses.



Replace Type Code with Subclass

You have an immutable type code that affects the behavior of a class.

Replace the type code with subclasses.



Primitive Obsession z1

What to Do

For simulated types: an integer type code stands in for a class.

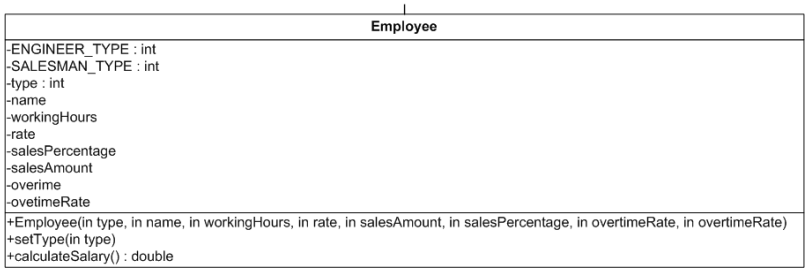
If the behavior of the object depends on the typecode and the type code changes during the lifetime of the object (or the class is subclassed already), ***Replace Type Code with Strategy***.

Replace Type Code with Strategy

```
void setType(int type){
    this.type = type;
}

double calculateSalary(){
    if (type == SALESMAN_TYPE){
        return workingHours*rate + salesPercentage*salesAmount;
    } else if (type == ENGINEER_TYPE)
        return workingHours*rate + overtime*overtimeRate;
}
```

Apply the Strategy pattern, i.e., encapsulate the typecode dependent behavior into separate objects with polymorphic interfaces and delegate calls to these objects

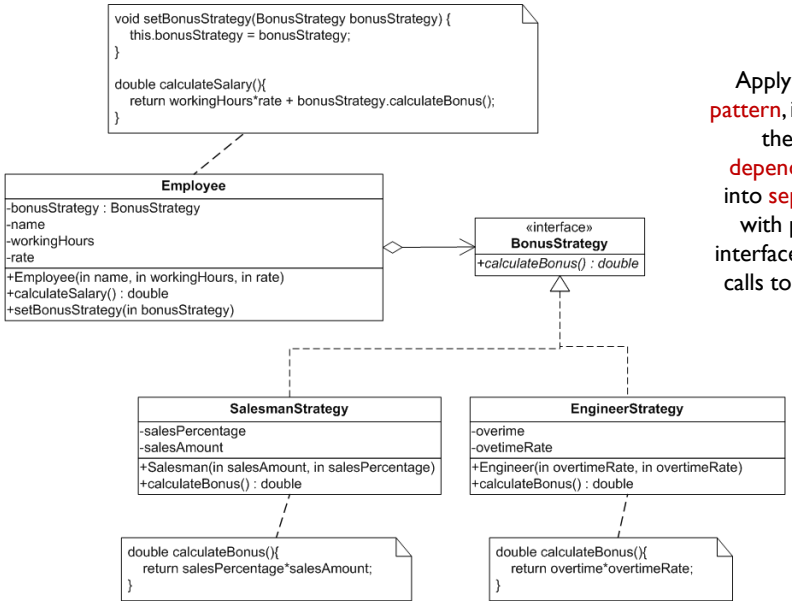


Slide 13

z1 use the example with some more details in the board

write down the if statements in the Employee class methods
zarras, 06/01/2020

Replace Type Code with Strategy



Apply the Strategy pattern, i.e., encapsulate the typecode dependent behavior into separate objects with polymorphic interfaces and delegate calls to these objects

Data Class/Container

Data Class/Container

Symptoms

The class consists only of public data members, or of simple getting and setting methods.

Causes

It's common for classes to begin like this: you realize that some data is part of an independent object, so you extract it out. But objects are about the commonality of *behavior, and these objects aren't developed enough to have much behavior yet.*

Data Class/Container

What to Do

Encapsulate Field to block direct access to the fields (allowing access only through getters and setters).

Encapsulate Field

There is a public field.

Make it private and provide accessors.

```
public String _name
```

```
private String _name;
public String getName() {return _name;}
public void setName(String arg) {_name = arg;}
```

Data Class/Container

What to Do

Encapsulate Collection to remove direct access to any collection-type fields.

Encapsulate Collection

A method returns a collection.

Make it return a read-only view and provide add/remove methods.

Data Class/Container

What to Do

Look at each client of the object. Almost invariably, you'll find clients accessing the fields and manipulating the results, when the class could do it for them.
(This is often a source of duplication, as many callers will tend to do the same things with the data.)
On the client, use ***Extract Method*** *to pull out the class-related code,* then ***Move Method*** *to put it over on the class.*

Data Class/Container

What to Do

After doing this a while, you may find that you have several similar methods on the class. Use *Rename Method, Extract Method,* ***Add Parameter, Remove Parameter****, etc. to harmonize signatures and remove duplication.*

Data Clump

Data Clump

Symptoms

The same two or three items frequently appear together in classes and parameter lists.
You also get a whiff of this when people declare some fields, then methods that work with those fields, then more fields and more methods, etc. (That is, there are groups of fields and methods together within the class.)

You may **see groups of field/parameter names** that start or end with **similar substrings**.

Causes

The values are typically part of some other entity, but no one has yet had the insight to realize that there's a **missing class**. Or sometimes, people know the object is missing, but think it's too small or unimportant to stand alone.

Data Clump

What to Do

1. If the values are fields in a class, use *Extract Class to pull them into a new class*.
2. If the values are together in method calls, *Introduce Parameter Object to extract* the new object.