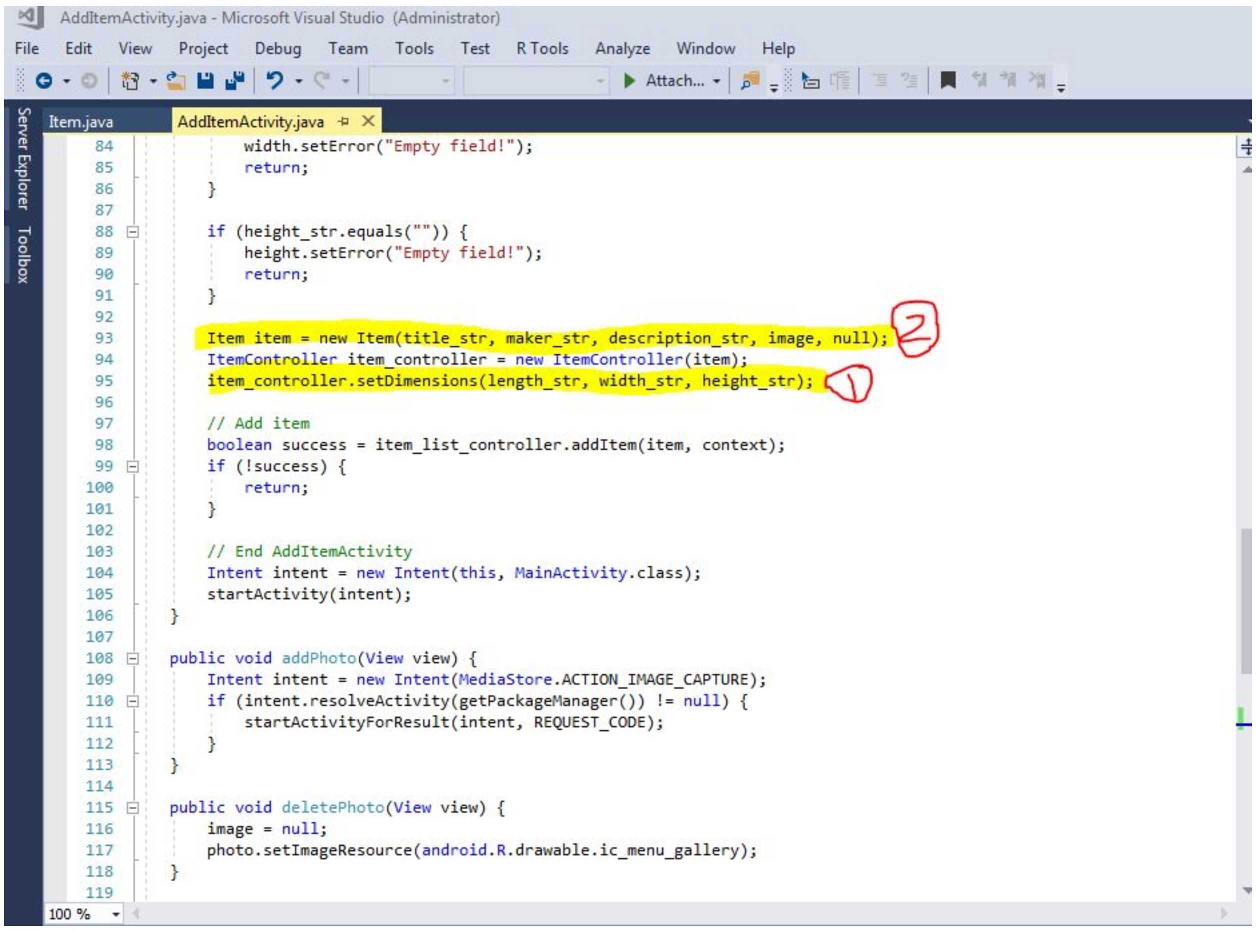
Peer-graded Assignment: Capstone Assignment 2.3 – Identify and Fix Code Smells

Name : Ahmed Khalifa

Course : Design Patterns - Coursera


AddItemActivity.java - Microsoft Visual Studio (Administrator)
File Edit View Project Debug Team Tools Test R Tools Analyze Window Help
Attach...
Server Explorer Toolbox
Item.java
AddItemActivity.java
2
1
100 %

```
            width.setError("Empty field!");
            return;
        }

        if (height_str.equals("")) {
            height.setError("Empty field!");
            return;
        }

        Item item = new Item(title_str, maker_str, description_str, image, null);
        ItemController item_controller = new ItemController(item);
        item_controller.setDimensions(length_str, width_str, height_str);

        // Add item
        boolean success = item_list_controller.addItem(item, context);
        if (!success) {
            return;
        }

        // End AddItemActivity
        Intent intent = new Intent(this, MainActivity.class);
        startActivity(intent);
    }

    public void addPhoto(View view) {
        Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
        if (intent.resolveActivity(getPackageManager()) != null) {
            startActivityForResult(intent, REQUEST_CODE);
        }
    }

    public void deletePhoto(View view) {
        image = null;
        photo.setImageResource(android.R.drawable.ic_menu_gallery);
    }
```

1- Data Clumps

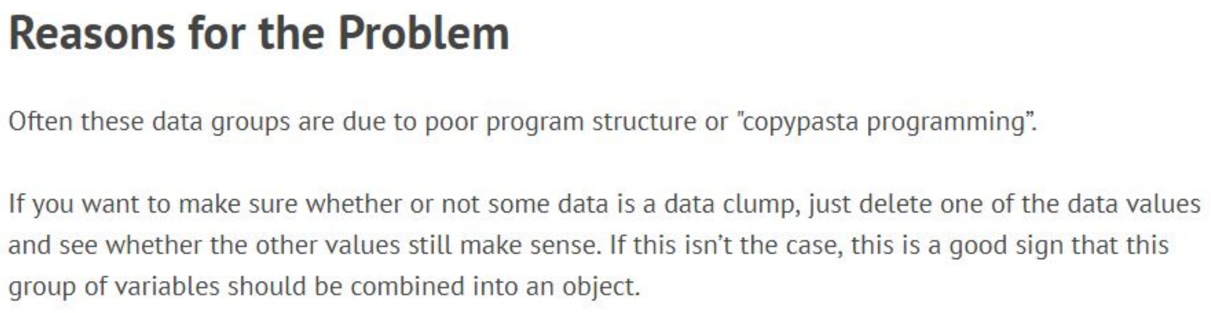
A very good Resource :

https://refactoring.guru/smells/data-clumps

1.1 Where The Code Smell :

- Class : AddItemActivity
- Method : saveItem

1.2 The Problem Of Data Clumps:

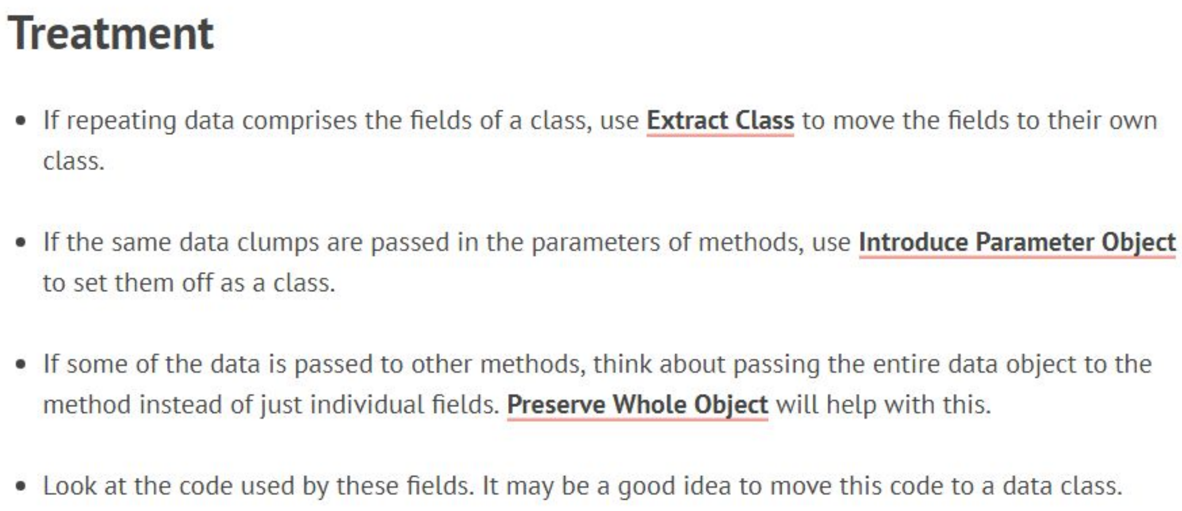


Reasons for the Problem

Often these data groups are due to poor program structure or "copypasta programming".

If you want to make sure whether or not some data is a data clump, just delete one of the data values and see whether the other values still make sense. If this isn't the case, this is a good sign that this group of variables should be combined into an object.

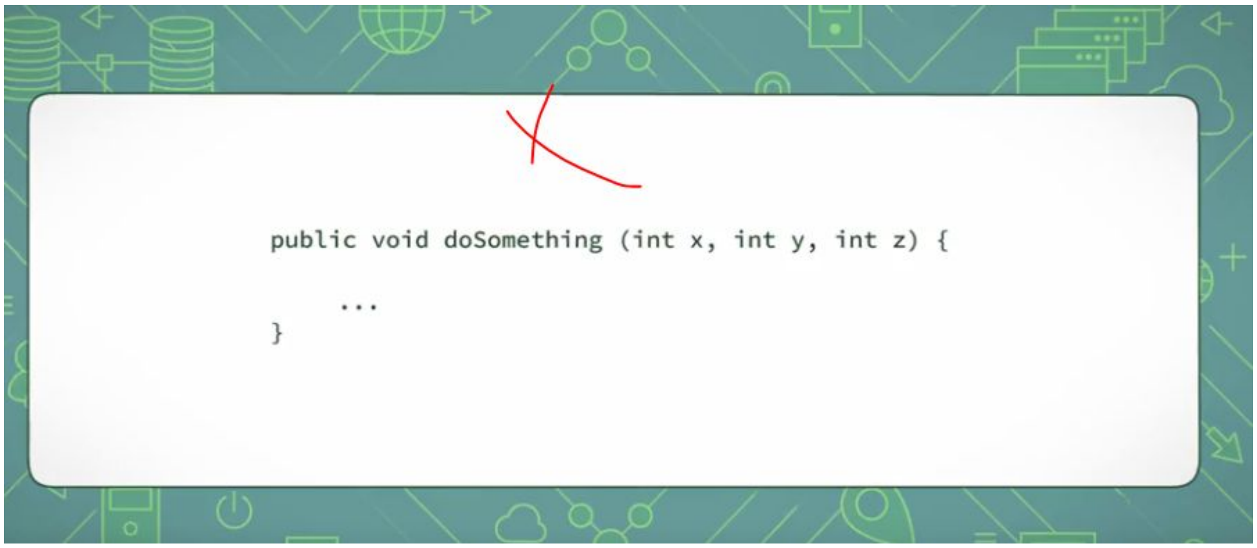
1.3 How To Refactor :



Treatment

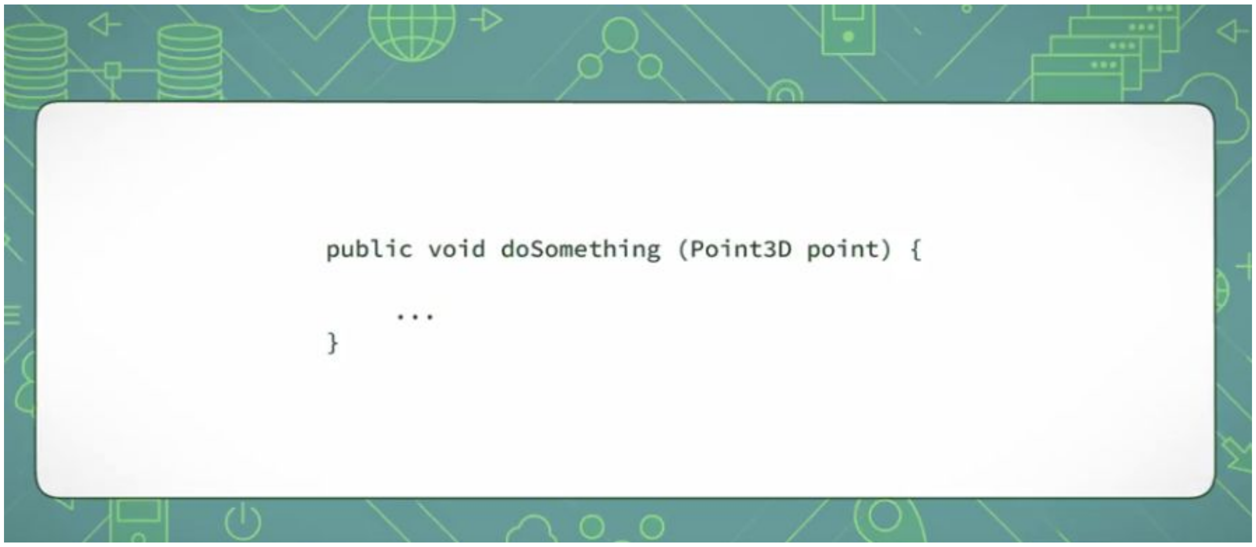
- If repeating data comprises the fields of a class, use **Extract Class** to move the fields to their own class.
- If the same data clumps are passed in the parameters of methods, use **Introduce Parameter Object** to set them off as a class.
- If some of the data is passed to other methods, think about passing the entire data object to the method instead of just individual fields. **Preserve Whole Object** will help with this.
- Look at the code used by these fields. It may be a good idea to move this code to a data class.

1-4 example of refactor code :

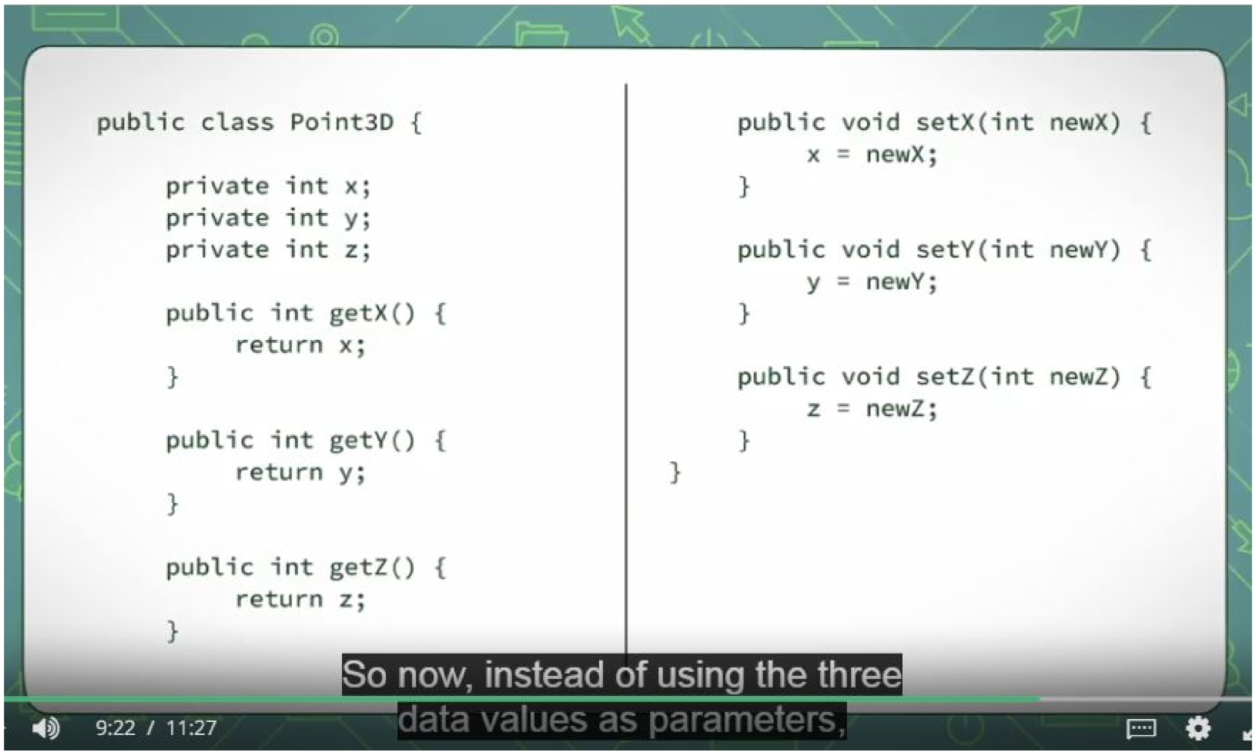


```
public void doSomething (int x, int y, int z) {

    ...
}
```



```java
public void doSomething (Point3D point) {
    ...
}
```



```java
public class Point3D {

    private int x;
    private int y;
    private int z;

    public int getX() {
        return x;
    }

    public int getY() {
        return y;
    }

    public int getZ() {
        return z;
    }
```

```java
    public void setX(int newX) {
        x = newX;
    }

    public void setY(int newY) {
        y = newY;
    }

    public void setZ(int newZ) {
        z = newZ;
    }
}
```

So now, instead of using the three data values as parameters,

Note : this refactor may be to cause another code smell data class , so we can add more methods to this class to perform related functionality.

2- Long Parameter List

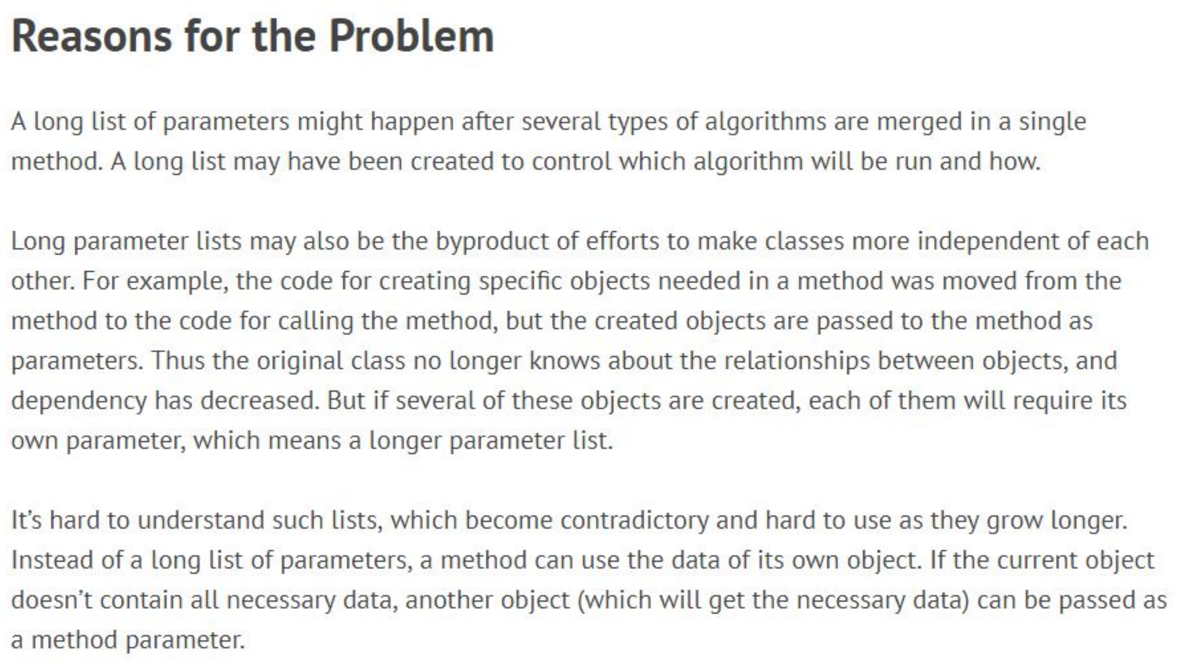
A very good Resource :

https://refactoring.guru/smells/long-parameter-list

1.1 Where The Code Smell :

- Class : AddItemActivity
- Method : saveItem

1.2 The Problem Of Code Smell :



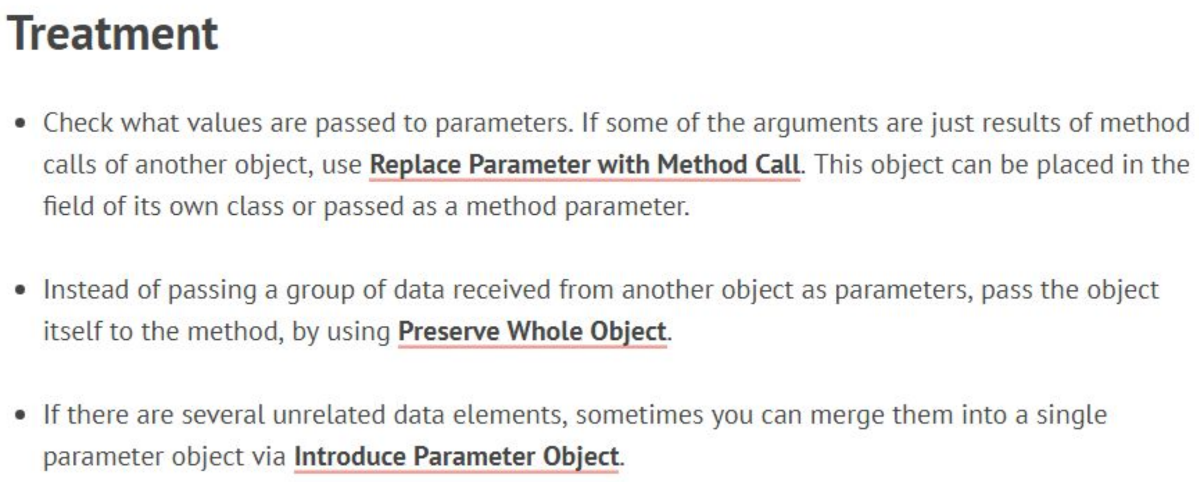
Reasons for the Problem

A long list of parameters might happen after several types of algorithms are merged in a single method. A long list may have been created to control which algorithm will be run and how.

Long parameter lists may also be the byproduct of efforts to make classes more independent of each other. For example, the code for creating specific objects needed in a method was moved from the method to the code for calling the method, but the created objects are passed to the method as parameters. Thus the original class no longer knows about the relationships between objects, and dependency has decreased. But if several of these objects are created, each of them will require its own parameter, which means a longer parameter list.

It's hard to understand such lists, which become contradictory and hard to use as they grow longer. Instead of a long list of parameters, a method can use the data of its own object. If the current object doesn't contain all necessary data, another object (which will get the necessary data) can be passed as a method parameter.

1.3 How To Refactor :



Treatment

- Check what values are passed to parameters. If some of the arguments are just results of method calls of another object, use **Replace Parameter with Method Call**. This object can be placed in the field of its own class or passed as a method parameter.

- Instead of passing a group of data received from another object as parameters, pass the object itself to the method, by using **Preserve Whole Object**.

- If there are several unrelated data elements, sometimes you can merge them into a single parameter object via **Introduce Parameter Object**.