**SOEN 384: Assignment 2**

**Team :**

Jordan Hubscher, ID : 27019696

Peter Boulos, ID: 26671187

1.

**Feature Envy**

**Package:**

[domain.](https://www.dropbox.com/sh/yl8uiqamtz7pl2h/AACUBHvc12VsUSocgZGyS1aAa?dl=0)mappers.files

**Classes:**

ClientMapper.java

**Method(s):**

update()

updateWithoutPhoto()

**Long Method**

**Package:**

[domain.](https://www.dropbox.com/sh/yl8uiqamtz7pl2h/AACUBHvc12VsUSocgZGyS1aAa?dl=0)mappers.files

**Classes:**

HouseMapper.java

ClientMapper.java

AppointmentMapper.java

FlagMapper.java

NoteMapper.java

**Method(s):**

insert()

map()

map()

readByBadge()

get()

addToHouseHold()

read()

2.

**Feature Envy**

public void update(Client client) {

try {

ClientTDG.update(client.getId(), client.getFirstName(),

client.getMiddleName(), client.getLastName(),

client.getDOB(), client.getSex().toString(),

client.getMedicare(), client.getMotherTongueId(),

client.getSecondLanguageId(), client.getPhoto(),

client.getThumb(),

client.getMaritalStatus().toString(),

client.getWorkStatus().toString(),

client.getExtraWorkStatusField(),

client.getCanadianStatus().toString(),

client.getExtraCanadianStatusField(),

client.getOrigin().toString(),

client.getExtraOriginField(),

client.getReferral().toString(),

client.getReferralExtra(),

client.getReason().toString(),

client.getReasonExtra(),

client.getRegistrationDate() );

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

As it is clearly visible, this method (along with nearly every other method in this class; such as the insert(), map(), and updateWithoutPhoto() methods to name a few) relies heavily on all attributes from the client object, when, ironically, in the map() method, all of the attributes are set to begin with. Either way the root issue here seems to be the fact that this mapper class (in this particular context) is rather useless and simply exists to pass around the client object’s instance variables like a hot potato.

**Long Method**

static Client map(ResultSet set) {

Client c = null;

try {

int ClientId = set.getInt(1);

String firstName = set.getString(2);

String middleName = set.getString(3);

String lastName = set.getString(4);

Date dob = new java.util.Date(set.getTimestamp(5).getTime());

String sex= set.getString(6);

String medicare = set.getString(9);

Relation relation = new RelationMapper().read(set.getInt(11));

String maritalStatus = set.getString(14);

String workStatus = set.getString(15);

String workStatusExtra = set.getString(16);

String canadaStatus = set.getString(17);

String canadaStatusExtra = set.getString(18);

String origin = set.getString(19);

String originExtra = set.getString(20);

String referral = set.getString(21);

String referralExtra = set.getString(22);

String reason = set.getString(23);

String reasonExtra = set.getString(24);

Date registeredDate = new java.util.Date(set.getTimestamp(25).getTime());

/\* Languages \*/

ResultSet querySet = LanguageTDG.readMother(ClientId);

Language mother = null;

Language second = null;

if (querySet.next()) {

mother = LanguageMapper.map(querySet);

}

querySet = LanguageTDG.readSecond(ClientId);

if (querySet.next()) {

second = LanguageMapper.map(querySet);

}

/\* Images \*/

BufferedImage image = null;

BufferedImage thumb = null;

InputStream in;

byte[] imageBytes;

imageBytes = set.getBytes(7);

if (imageBytes != null) {

in = new ByteArrayInputStream(imageBytes);

image = ImageIO.read(in);

}

byte[] thumbBytes = set.getBytes(8);

if (thumbBytes != null) {

in = new ByteArrayInputStream(thumbBytes);

thumb = ImageIO.read(in);

}

c = new Client(ClientId, firstName, middleName, lastName, image, thumb, dob, sex, medicare, mother, second,relation, maritalStatus, workStatus, workStatusExtra, canadaStatus, canadaStatusExtra, origin, originExtra, referral, referralExtra, reason, reasonExtra, registeredDate);

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return c;

}

As one may see here, this method is particularly long and does in fact, more than one thing. However, the main why is it lexically large is mostly due to the reasons from the previous listed method (i.e. feature envy, excessive variables). However, this method is still taking alternate action based on the state of thumbBytes/thumb variables. This may be extracted to an external method.

3.

**Feature Envy**

Instead of passing a bunch of arguments to a given class method-where no further updating, processing is being held, mind you-that will just end up being passed to another class method, just pass THAT object to the final class method. So, in this case, a Client object is passed to the update() method in the ClientMapper class. Then, all the instance variables of the Client object are being passed as arguments to the update() method from the ClientTDG class, when in reality, all this trouble and disorganization can be saved by “moving” this method to the class that actually will be using it, in this case, ClientTDG. And instead of taking all the arguments of a Client object as arguments, it will take simply a Client object instead and do whatever calculations it must from there. No need to force some sort of “real-world” structure where it doesnt practically apply.

**Long Method**

Instead of having that both conditionals (i.e. **if** (imageBytes != null) and **if** (thumbBytes != null)) handle their byte array identically, an external/extracted method would work quite nicely here. A method called setByteArray() and it would take a Byte Array and a BufferedImage as arguments, and return a ByteArrayInputStream. It would then be identical code (from the block of either conditionals.) It would check if the Byte array were null, and then if it is not, it would then initialize the ByteArrayInputStream and then read in the bytes into the BufferedImage. Last step, return the initialized the ByteArrayInputStream, otherwise it would return null.

ByteArrayInputStream setByteArray(BufferedImage image, Byte[] byteArr) {

if (byteArr != null) {

ByteArrayInputStream in = new ByteArrayInputStream(byteArr);

image = ImageIO.read(in);

}

}