JDBC lab

Object-relational mismatch

- The relational and object oriented models are quite different

Object-relational mismatch

- The relational and object oriented models are quite different
- When making an app that speaks to a database we need to find a way to ***bridge this gap***

Object-relational mismatch

- The relational and object oriented models are quite different
- When making an app that speaks to a database we need to find a way to ***bridge this gap***
- How do we structure an application? How do we talk to a database?

Object-relational mismatch

- The best thing to go about this is to start by thinking ***what our application needs***

Object-relational mismatch

- The best thing to go about this is to start by thinking ***what our application needs***
- Let the application requirements guide you!

Tessiland - requirements

- There’s an ***home page,*** showing all of our products’ previews

Tessiland - requirements

- There’s an ***home page,*** showing all of our products’ previews
- There’s a ***product page,*** showing details about a specific product

Tessiland - previews

A product preview has to show the product's name and tags:

```
public class ProductPreview {
  public final int code;
  public final String name;
  public final List<Tag> tags;
}

public final class Tag {
  public final String name;
}
```

Tessiland - product page

A product page has to show the product's name, description and composition:

```
public class Product {
  public final int code;
  public final String name;
  public final String description;
  public final Map<Material, Float> composition;
}

public final class Material {
  public final int code;
  public final String description;
}
```

Tessiland - bridging the gap

- Now that we now what kind of objects our application needs we can think how to make queries to the database

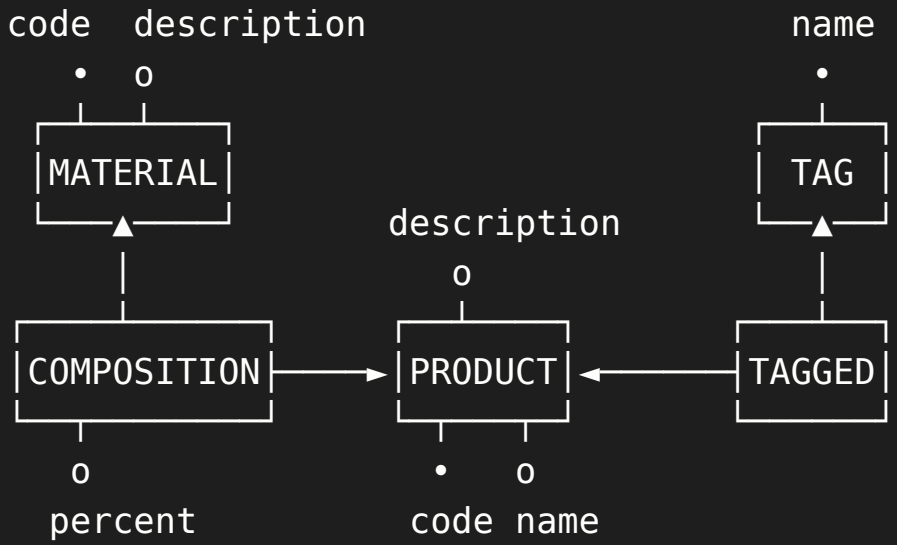
Tessiland - bridging the gap

- Now that we now what kind of objects our application needs we can think how to make queries to the database
- We'll define ***DAOs (Database Access Objects)*** that can help us carry out those queries

Tessiland - building `ProductPreviewS`

```
public class ProductPreview {
  public class DAO {
    // The only operation we ever need to do with
    // product previews is list them to fill the home
    // page, there's no need to do anything else!
    public static List<ProductPreview> list(
      Connection connection
    ) { ... }
  }
}
```

Tessiland - building `ProductPreviewS`



Tessiland - building `ProductPreview`s

```
public class ProductPreview {
  public class DAO {
    public static List<ProductPreview> list(
      Connection connection
    ) {
      // 1. Run a query to read all products
      //    a. Only read what we actually need!
      //       that is the product's id and name.
      //       That's all we need to make a preview
      // 2. As we iterate through products we'll
      //    have to fetch their tags as well
    }
  }
}
```

Tessiland - building `ProductPreview`s

```
public class ProductPreview {
  public class DAO {
    public static List<ProductPreview> list(
      Connection connection
    ) {
      // 1. Run a query to read all products
      //    a. Only read what we actually need!
      //       that is the product's id and name.
      //       That's all we need to make a preview
      // 2. As we iterate through products we'll
      //    have to fetch their tags as well
    }
  }
}
```

Try implementing the missing pieces

- Implement the queries needed to load a product page
  - You'll need to fetch a specific product from the database
  - You'll need to build a map that represents the product's composition
- Play with the application, the project already has a simple GUI
- Try adding even more queries: adding new tags, new products, removing tags...