METODY PROGRAMOWANIA

Wzorzec Repository

21 października 2017

Krzysztof Pawłowski
kpawlowski@pjwstk.edu.pl

"Każdy wzorzec opisuje problem, który ciągle pojawia się w naszej dziedzinie, a następnie określa zasadniczą część jego rozwiązania w taki sposób, by można było zastosować je nawet milion razy, za każdym razem w nieco inny sposób."

A Patterns Language (1977)
— Alexander Christopher

CZYM JEST WZORZEC PROJEKTOWY?

- sprawdzone w praktyce rozwiązanie powtarzalnych problemów projektowych

- powiązania i zależności pomiędzy klasami i obiektami

- jest opisem rozwiązania a nie jego implementacją

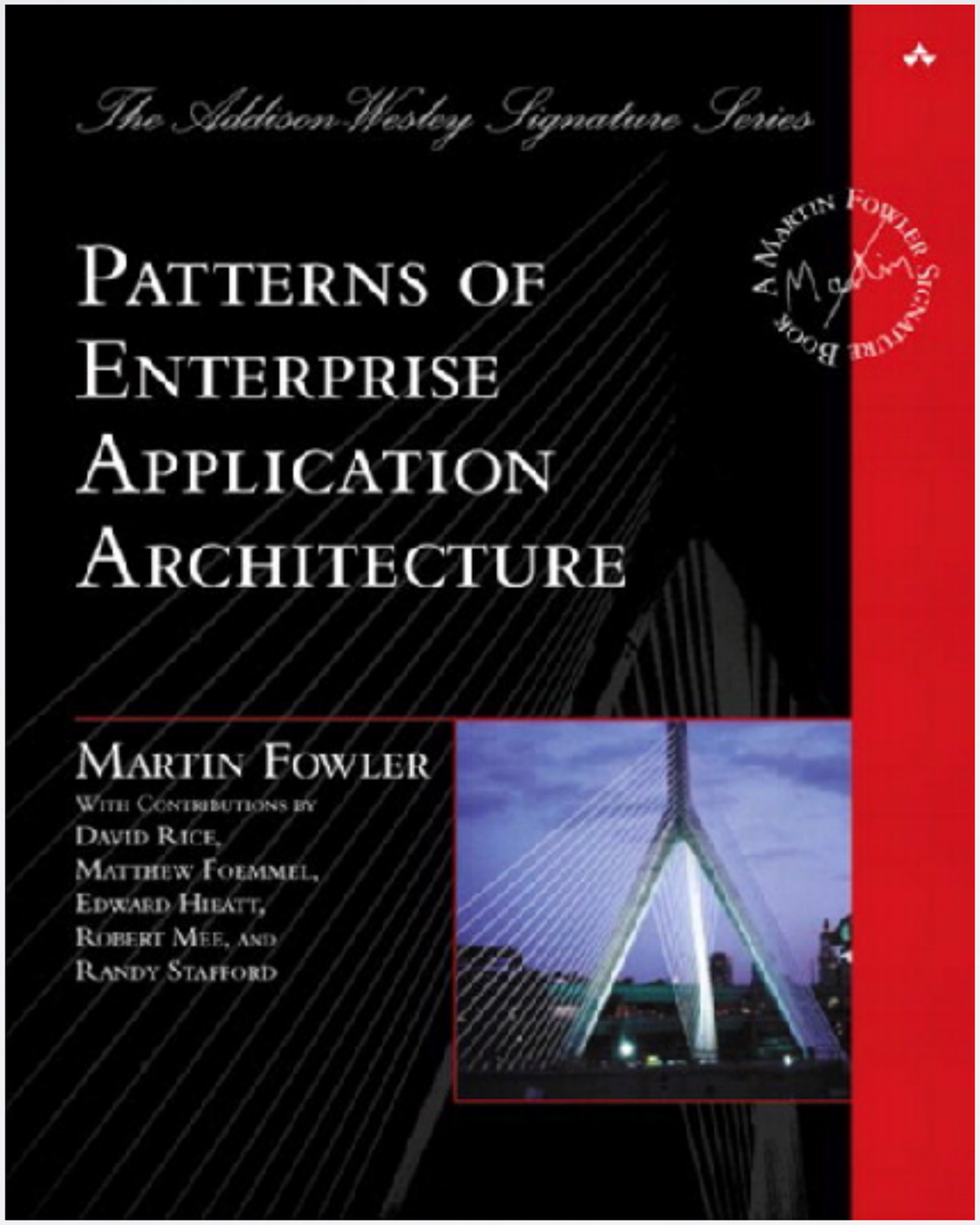
POCZĄTKI WZORCÓW PROJEKTOWYCH

Design Patterns: Elements of Reusable Object-Oriented Software, 1995

autorzy: Gamma, Help, Johnson, Vlissides

- Katalog 23 wzorców projektowych
- Pokazanie zastosowania wzorców projektowych w dziedzinie projektowania oprogramowania

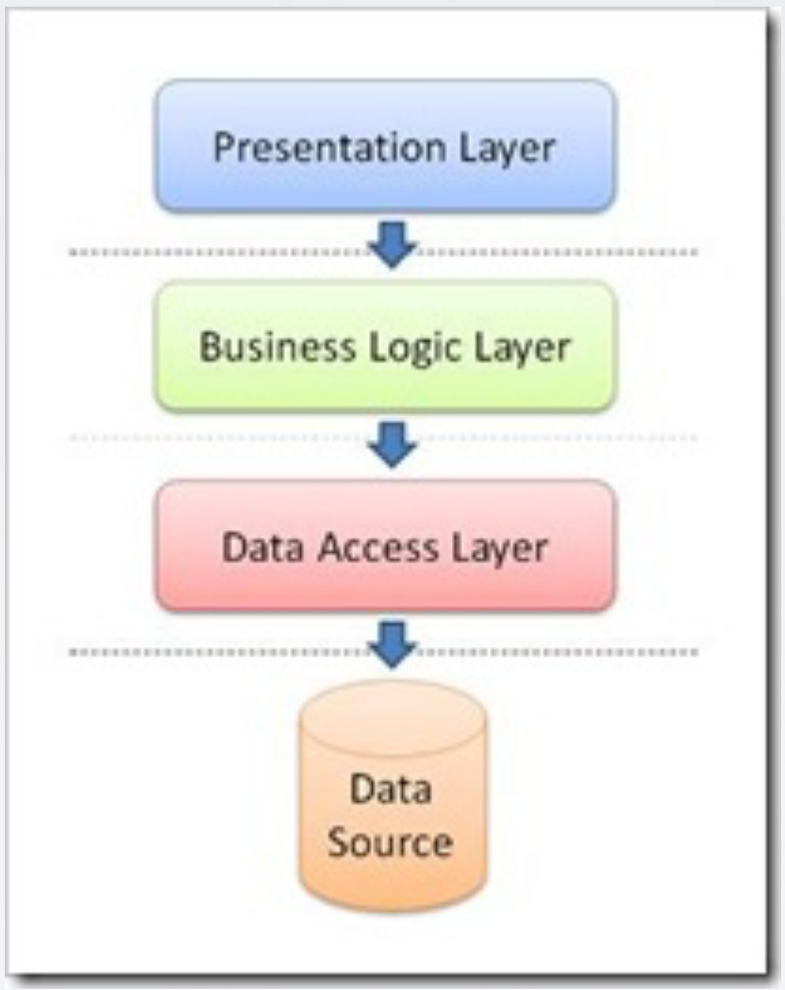
GDZIE SZUKAĆ OPISÓW WZORCÓW ARCHITEKTONICZNYCH?



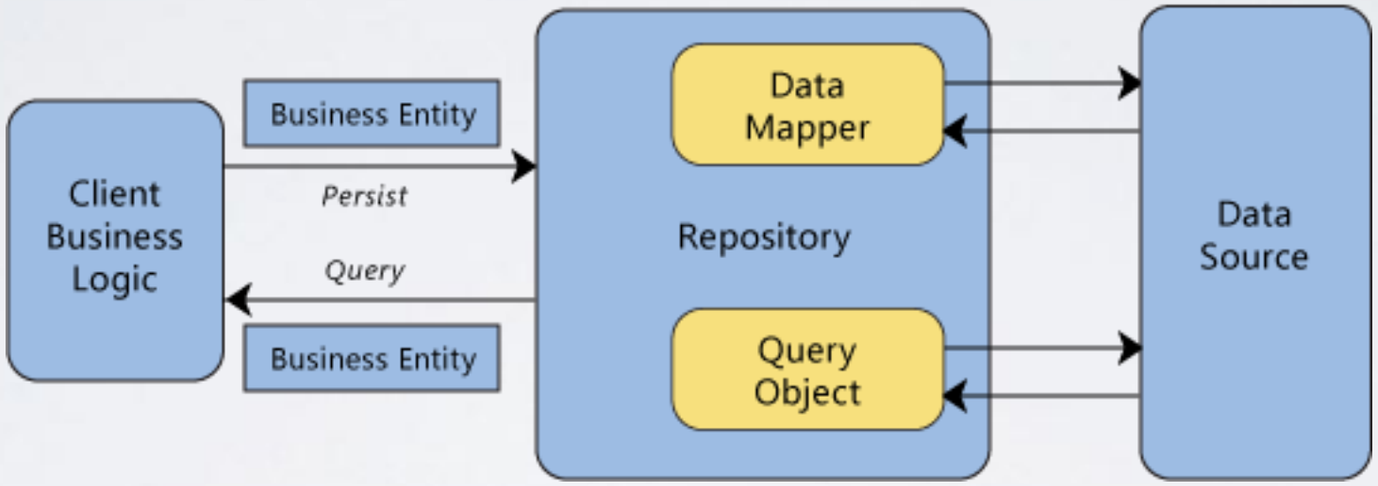
WZORZEC REPOSITORY - MOTYWACJA

- zasada DRY (Don't Repeat Yourself)
- zasada KISS (Keep It Simple, Stupid)
- łatwość zmiany bazy danych
- łatwość testowania

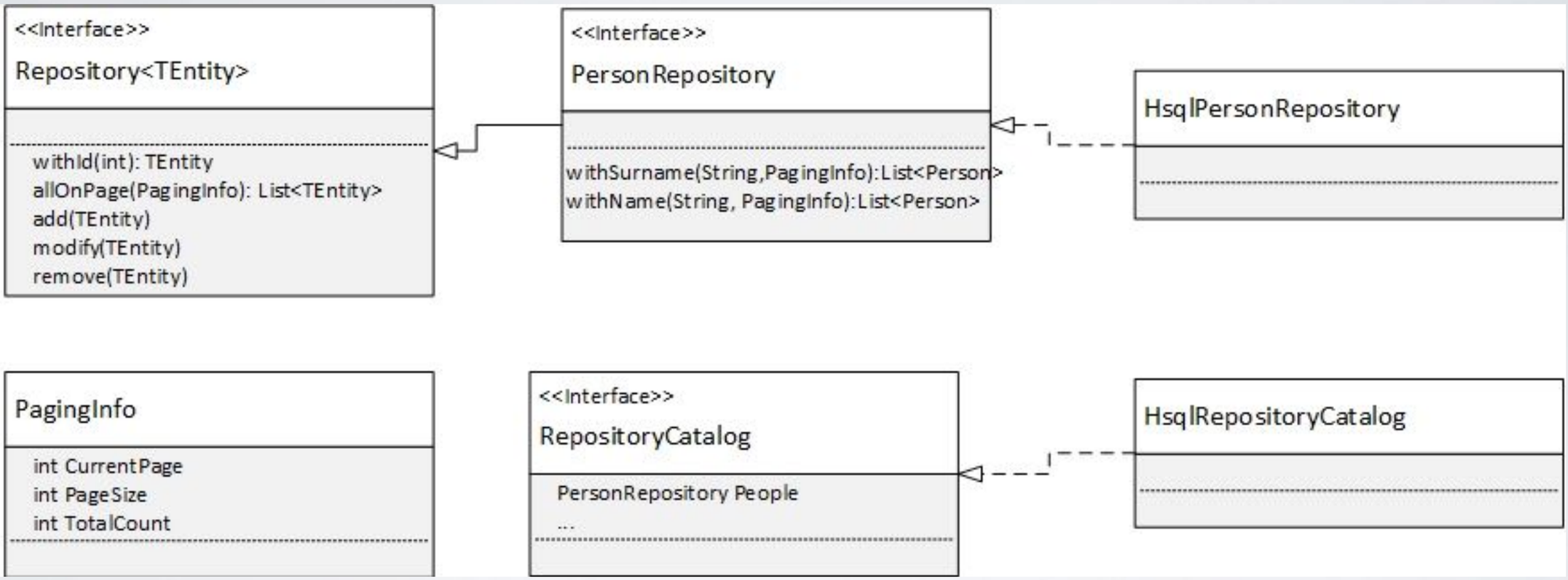
ARCHITEKTURA 3-WARSTWOWA



WZORZEC REPOSITORY



WZORZEC REPOSITORY - PRZYKŁAD

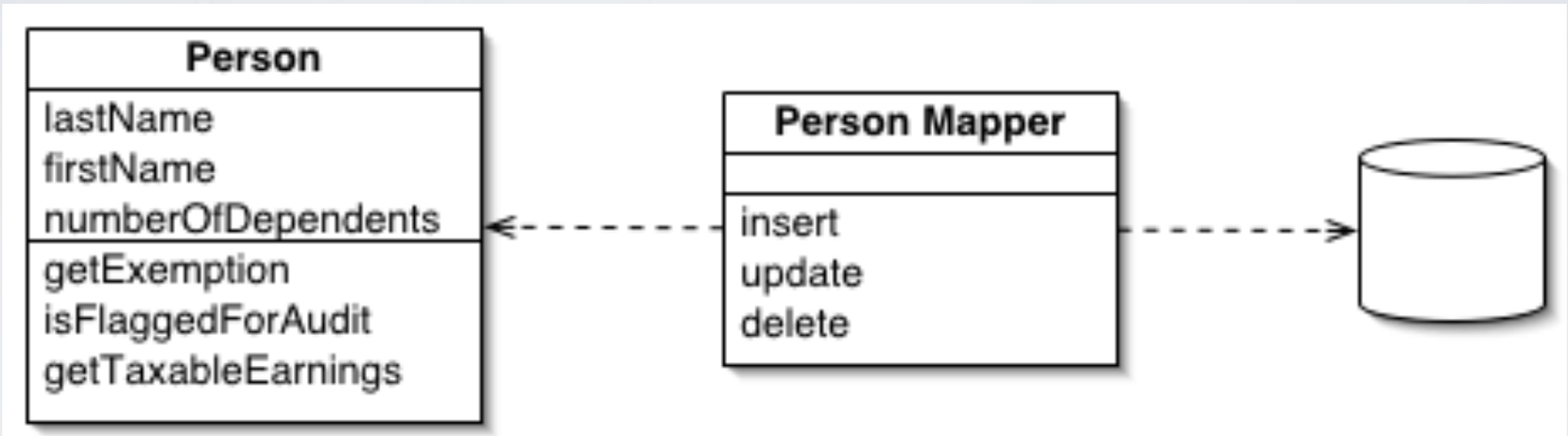


WZORZEC REPOSITORY - PRZYKŁAD

- https://github.com/krzysztof-pawlowski/MPR-repository-pattern

(branch *to_refactor*)

WZORZEC DATA MAPPER



- "tłumaczy" tabele bazodanowe na klasy modelu
- pozwala na niezależne rozwijanie modelu i schematu bazy danych

WZORZEC DATA MAPPER - PRZYKŁAD

klasa:

```
class Person {
  private String lastName;
  private String firstName;
  private int numberOfDependents;
}
```

tabela:

```
create table people (ID int primary key,
lastname varchar, firstname varchar,
number_of_dependents int)
```

WZORZEC DATA MAPPER - PRZYKŁAD

```
class PersonMapper...

  protected String findStatement() {
    return "SELECT " + COLUMNS + "  FROM people" + "
WHEREid=?";
  }

  public static final String COLUMNS = "id, lastname,
firstname, number_of_dependents";

  public Person find(Long id) {
    return (Person) abstractFind(id);
  }
```

WZORZEC DATA MAPPER - PRZYKŁAD

```
class AbstractMapper<T>...
  protected Map<Long, T> loadedMap = new HashMap();
  abstract protected String findStatement();
  protected T abstractFind(Long id) {
    T result = loadedMap.get(id);
    if (result != null) return result;
  PreparedStatement findStatement = null;
  try {
    findStatement = DB.prepare(findStatement());
    findStatement.setLong(1, id.longValue());
    ResultSet rs = findStatement.executeQuery();
    rs.next();
    result = load(rs);
    return result;
  } catch (SQLException e) {
    throw new ApplicationException(e);
  } finally {
    DB.cleanUp(findStatement);
  }
}
```

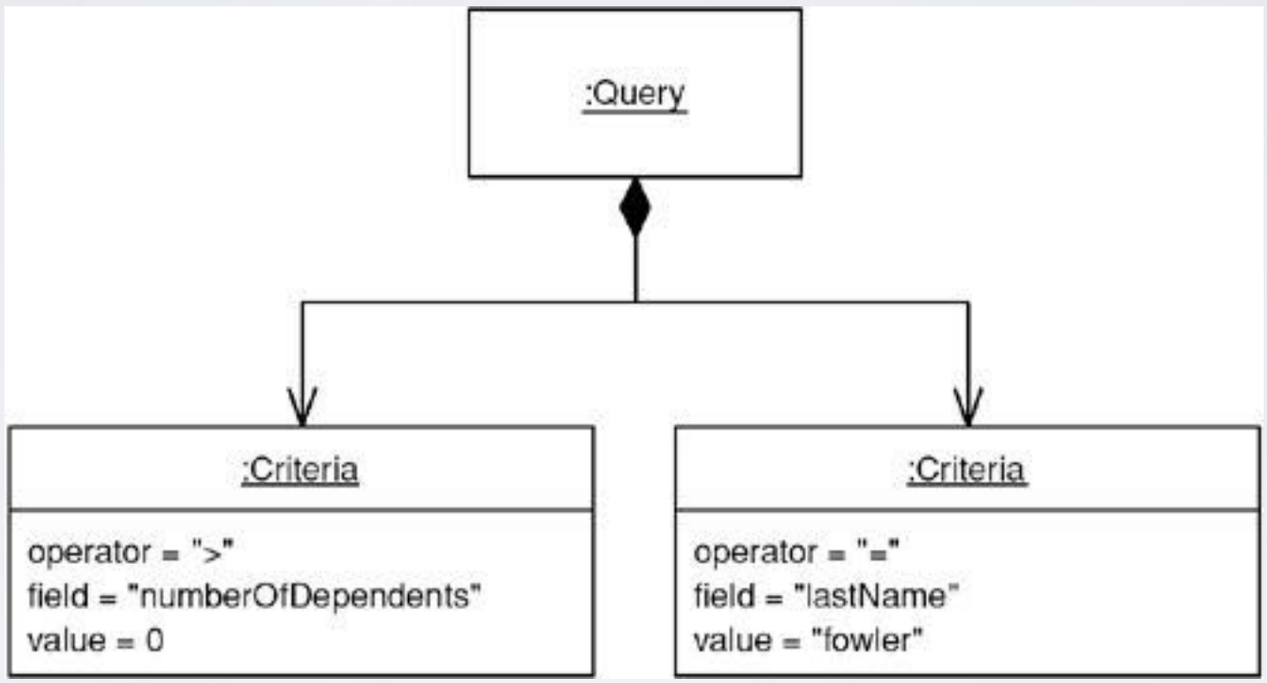
WZORZEC DATA MAPPER - PRZYKŁAD

```
class AbstractMapper<T>...

  protected T load(ResultSet rs) throws SQLException {
     Long id = new Long(rs.getLong(1));
     if (loadedMap.containsKey(id))
       return (DomainObject)  loadedMap.get(id);
    T result = doLoad(id, rs);
    loadedMap.put(id, result);
    return result;
}
abstract protected T doLoad(Long id, ResultSet rs)
  throws SQLException;
```

WZORZEC DATA MAPPER - PRZYKŁAD

```
class PersonMapper extends
AbstractMapper<Person>...

  protected Person doLoad(Long id,
    ResultSet rs) throws SQLException {
    String lastNameArg = rs.getString(2);
    String firstNameArg = rs.getString(3);
    int numDependentsArg = rs.getInt(4);
    return new Person(id, lastNameArg,
      firstNameArg, numDependentsArg);
}
```

WZORZEC QUERY OBJECT



- obiekt reprezentujący zapytanie bazodanowe

WZORZEC QUERY OBJECT - PRZYKŁAD

```
class QueryObject...

  private Class klass;
  private List criteria = new
    ArrayList<Criteria>();

class Criteria...

  private String sqlOperator;
  protected String field;
  protected Object value;
```

WZORZEC QUERY OBJECT - PRZYKŁAD

```
class Criteria...

 public static Criteria greaterThan(String fieldName, int value) {
   return Criteria.greaterThan(fieldName, new Integer(value));
 }
```

```
public static Criteria greaterThan(String fieldName, Object
  value){
  return new Criteria(" > ", fieldName, value);
}

private Criteria(String sql, String field, Object value) {
  this.sqlOperator = sql;
  this.field = field;
  this.value = value;
}
```

WZORZEC QUERY OBJECT - PRZYKŁAD

```
class Person...

  private String lastName;
  private String firstName;
  private int numberOfDependents;
```

```
QueryObject query = new QueryObject(Person.class);
Criteria c = Criteria
  .greaterThan("numberOfDependents", 0);
query.addCriteria(c);
query.generateSql();
```