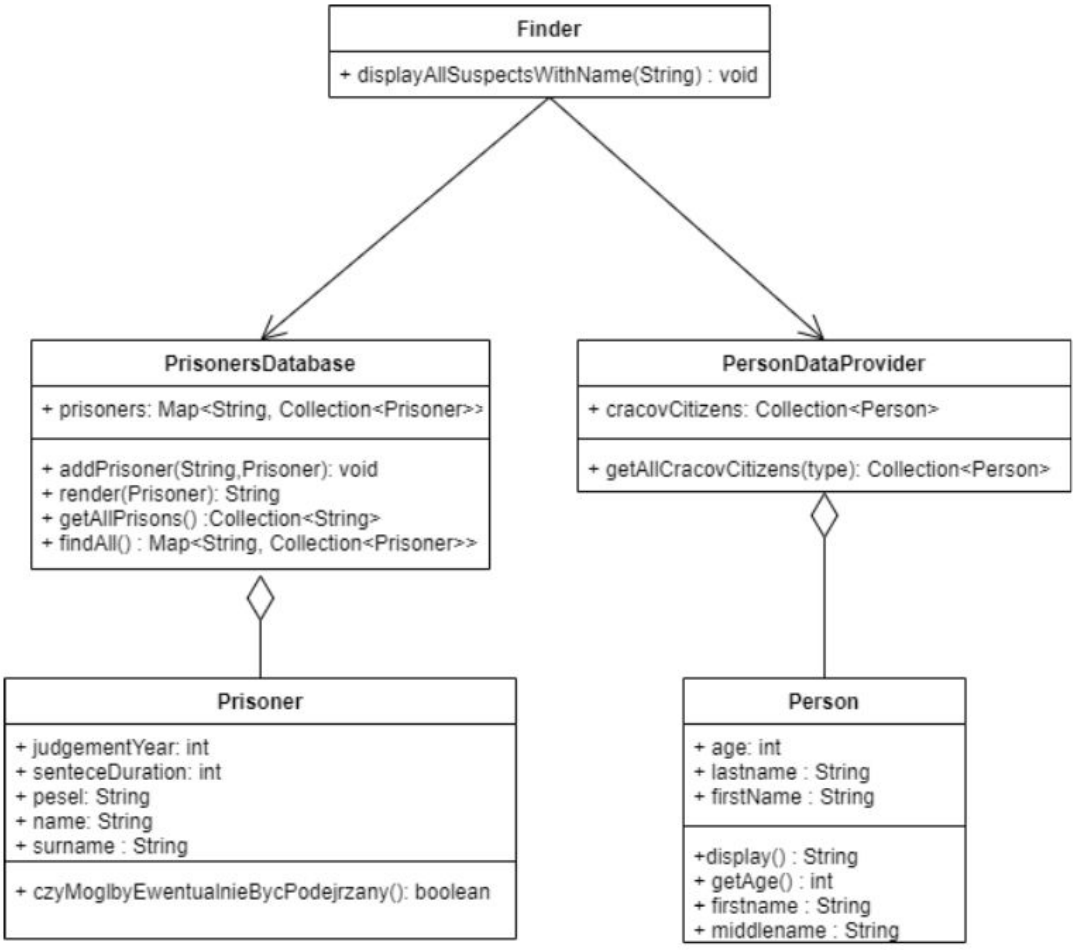
Projektowanie obiektowe
Laboratorium 4 Kwiecień 2020
Refaktoryzacja

Radosław Kopeć 305333

Zadanie 1



xxx() -> yyy() będzie oznaczać dla mnie zmianę nazwy

Finder
Jeden konstruktor byłby czytelniejszy

PirsonersDatabase:
przenieść metodę render() do Pirsoner i zmienić na toString()
zamiast findAll() -> getAllPirsoners()
zmiana nazwy klasy na PrisonerDataProvider

W Person

Zmiany nazw:

lastname -> lastName

firstname -> firstName

firstname() -> getFirstName

middlename() -> getLastName

W Prisoner

czyMoglbyEwentualnieBycPodejrzany() -> isInPrison()

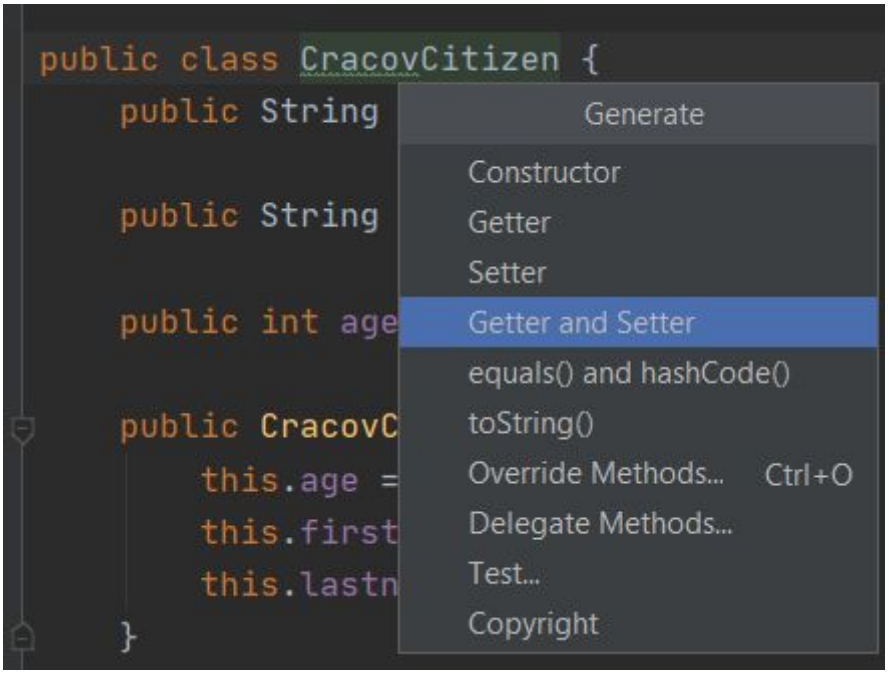
na dobrą sprawę Prisoner mógłby dziedziczyć po Person, aby uzyskać dodatkowe dane

PersonDataProvider

getAllCracovCitizens() -> getCracovCitizens()

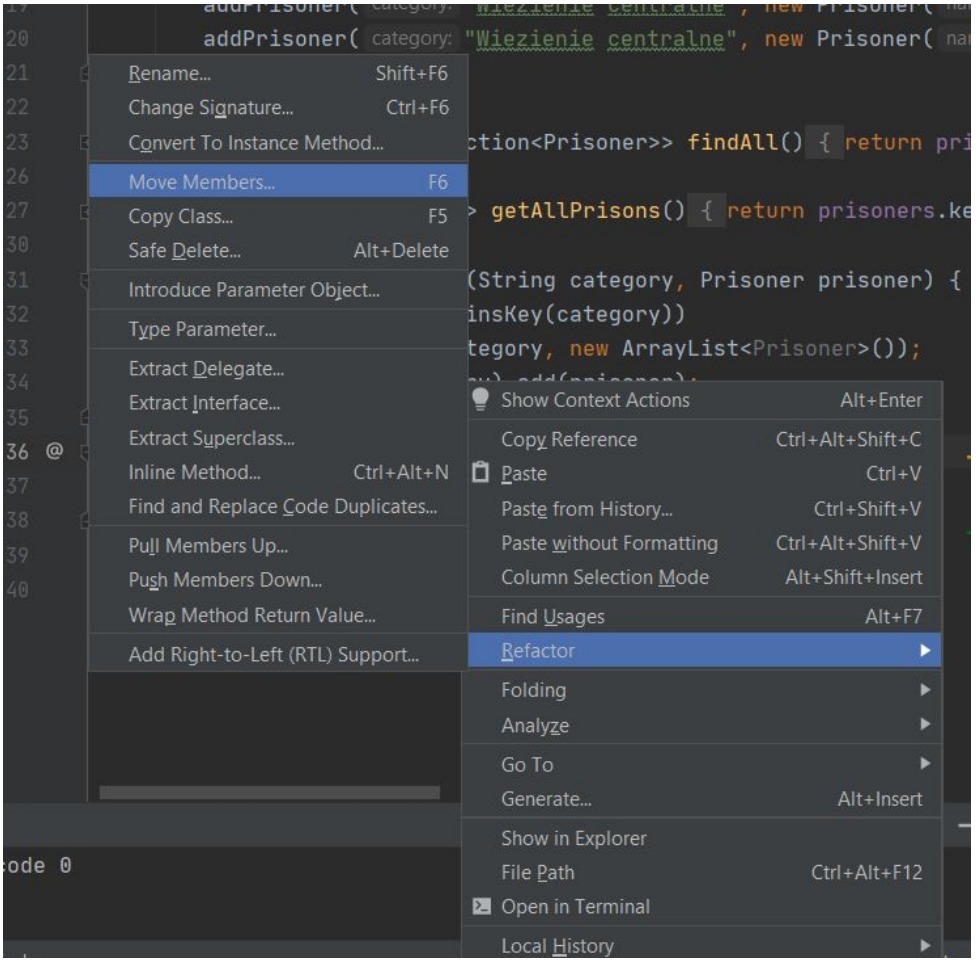
Zadanie 2

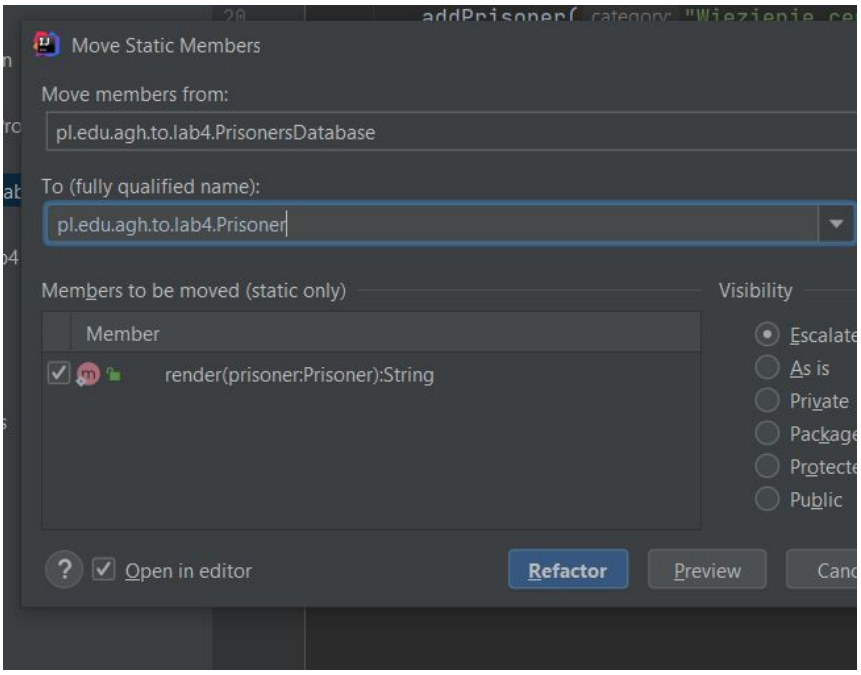
1. prywatne pola i metody dostpowe



2. statyczne metody w złych miejscach

Przenosimy metode render z PirsonerDatabase do display w Prisoner





W Prisoner

```
    }

    public static String render(Prisoner prisoner) {
        return prisoner.getName() + " " + prisoner.getSurname();
    }
```

```
@   public static String rer
        return prisoner.getN
    }

    public String getPesel()

    public boolean isJailedN
```

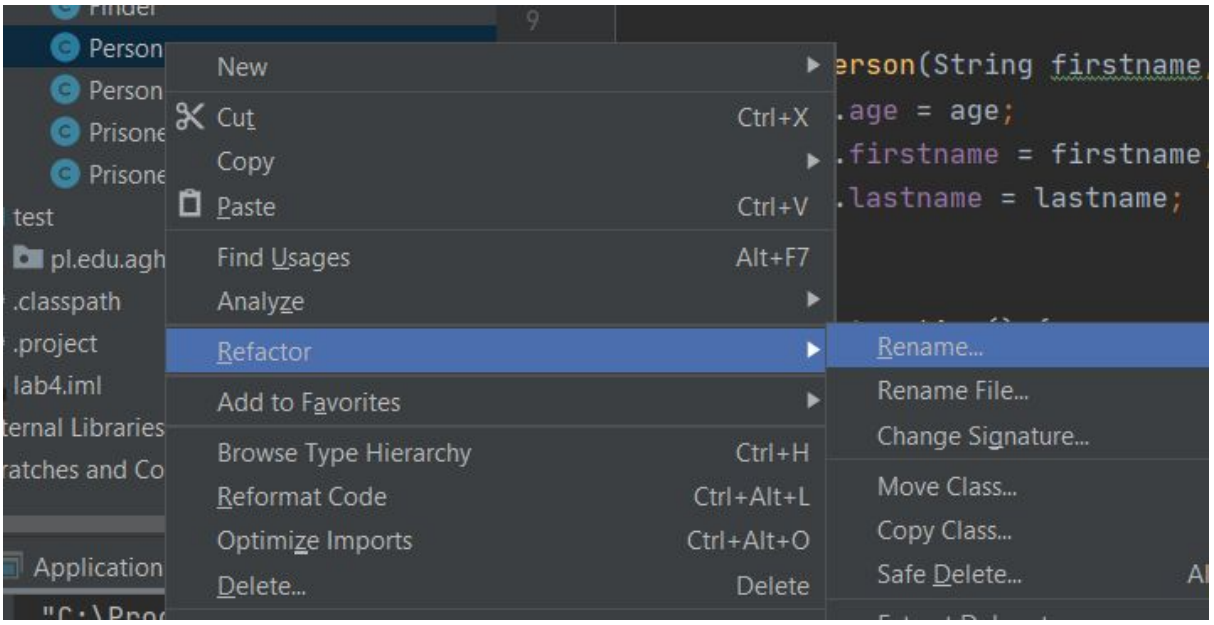
Show Context Actions Alt+Enter
Copy Reference Ctrl+Alt+Shift+C
Paste Ctrl+V
Paste from History... Ctrl+Shift+V
Paste without Formatting Ctrl+Alt+Shift+V
Column Selection Mode Alt+Shift+Insert
Find Usages Alt+F7
Refactor
Folding
Rename... Shift+F6
Change Signature... Ctrl+F6

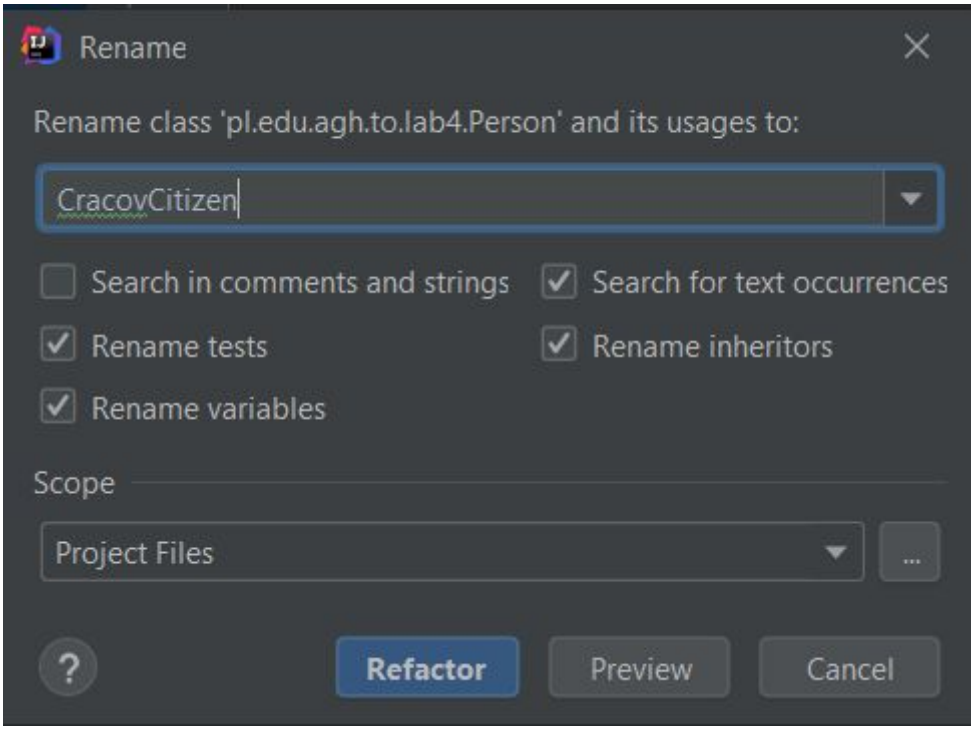
```
    }

    public static String display(Prisoner pri
        return prisoner.getName() + " " + pri
    }
```

3. zbyt długie i enigmatyczne nazwy metod

Refactor -> Reaname gdzie się tylko da

Zmiana Peron na CracovCitizen





Krok 3

Zaproponuj generalizację klas Person i Prisoner.

Mogą one implementować wspólny interfejs np. Person który wymusz implementacje ich wspólmnych cech takich jak imię, nazwisko, wyświetlanie.

```
package pl.edu.agh.to.lab4;

public interface Suspect {
    String getName();
    String getSurname();
    String display();
}
```

W Prisoner

```
public class Prisoner implements Suspect{
    private final int judgementYear;
```

```
public String display() {
    return this.getName() + " " + this.getSurname();
}
```

W CracovCitizen

```
public class CracovCitizen implements Suspect {
    public String firstname;
```

W Finder

```
for (Prisoner n : suspectedPrisoners) {
    System.out.println(n.display());
}
```

Krok 4

Dodanie nowego interfejsu

```
public interface SuspectAggreagate {
    Iterator<? extends Suspect> iterator();
}
```

Zmiany w PrisonersDatabase

```
public class PrisonersDatabase implements SuspectAggreagate {
```

```
    @Override
    public Iterator<Suspect> iterator() {
        return new FlatIterator(prisoners);
    }
```

Zmiany w PersonDataProvider

```
public class PersonDataProvider implements SuspectAggreagate {
```

```
@Override
public Iterator<? extends Suspect> iterator() {
    return cracovCitizens.iterator();
}
```

FlatIterator:

```java
public class FlatIterator implements Iterator {

    private Map<String, Collection<Prisoner>> prisoners = new HashMap<~>();
    private int iterationState;
    private Prisoner currentPrisoner;
    FlatIterator(Map<String, Collection<Prisoner>> prisoners){
        this.prisoners = prisoners;
        iterationState = 0;

    }
    @Override
    public boolean hasNext() {
        int length = 0;
        for (Collection<Prisoner> prisonerCollection : prisoners.values()) {
            length += prisonerCollection.size();
       }
        return iterationState < length;
    }
```

```java
@Override
public Prisoner next() {
    boolean returnInNextInteration = false;

    if(iterationState == 0){
        returnInNextInteration = true;
    }
    for (Collection<Prisoner> prisonerCollection : prisoners.values()) {
        for (Prisoner prisoner : prisonerCollection) {
            if (returnInNextInteration) {
                currentPrisoner = prisoner;
                iterationState ++;
                return currentPrisoner;
            }

            if (prisoner == currentPrisoner) {
                returnInNextInteration = true;
            }


        }
    }
    return null;
}
```

Modyfikacja Suspect na klasę abstrakcyjną

```java
public abstract class Suspect {

    protected String firstName;
    protected String lastName;
    protected int Age;

    public Suspect(String firstName, String lastName, int age) {
        this.firstName = firstName;
        this.lastName = lastName;
        Age = age;
    }
    public Suspect(String firstName, String lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
    }
    public String firstName(){
        return this.firstName;
    }
    public String lastName(){
        return this.lastName;
    }
    public String display() {
        return this.firstName() + " " + this.lastName();
    }
    boolean canBeAccused(){
        return true;
    }

    public int getAge() {
        return Age;
    }
}
```

Prisoner:

```java
public class Prisoner extends Suspect {

    private final int judgementYear;

    private final int senteceDuration;

    private final String pesel;


    public Prisoner(String name, String surname, String pesel, int judgementYear, int sentenceDuration) {
        super(name,surname);
        this.pesel = pesel;
        this.judgementYear = judgementYear;
        this.senteceDuration = sentenceDuration;
    }

    public String getPesel() { return pesel; }

    public boolean isJailedNow() {
        return judgementYear + senteceDuration >= getCurrentYear();
    }

    private static int getCurrentYear() {
        return Calendar.getInstance().get(Calendar.YEAR);
    }

    @Override
    public boolean canBeAccused(){
        return !isJailedNow();
    }
```

CracovCitizen:

```java
package pl.edu.agh.to.lab4;

public class CracovCitizen extends Suspect {

    public CracovCitizen(String firstname, String lastname, int age) {
        super(firstname,lastname,age);
    }

    @Override
    public boolean canBeAccused(){
        return this.Age > 18;
    }
}
```

Krok 5

Dodanie klasy CompositeAggregate:

```java
public class CompositeAggregate implements SuspectAggreagate{

    Collection<SuspectAggreagate> aggreagates = new ArrayList<>();
    Collection<Iterator<? extends Suspect>> iterators = new ArrayList<>();

    public CompositeAggregate(){}

    public void addAggregate(SuspectAggreagate a){
        aggreagates.add(a);
        iterators.add(a.iterator());
    }
```

```java
    @Override
    public Iterator<? extends Suspect> iterator() {
        Iterator<? extends Suspect> iterator = new Iterator<Suspect>() {
            @Override
            public boolean hasNext() {
                boolean has = false;
                for(Iterator<? extends Suspect> i : iterators){
                    if(i.hasNext()){
                        has = true;
                        break;
                    }
                }
                return has;
            }

            @Override
            public Suspect next() {
                if(!hasNext())return null;
                for(Iterator<? extends Suspect> i : iterators){
                    if(i.hasNext()){
                        return  i.next();
                    }
                }
                return null;
            }
        };
        return iterator;
    }
}
```

Zmiana Findera:

```java
package pl.edu.agh.to.lab4;

import java.util.ArrayList;
import java.util.Collection;
import java.util.Iterator;
import java.util.Map;

public class Finder {


    private CompositeAggregate compositeAggregate = new CompositeAggregate();

    public Finder(Collection<CracovCitizen> allCracovCitizens, Map<String, Collection<Prisoner>> allPrisoners) {
        PersonDataProvider personDataProvider = new PersonDataProvider();
        personDataProvider.setCracovCitizens(allCracovCitizens);
        PrisonersDatabase prisonersDatabase = new PrisonersDatabase();
        prisonersDatabase.setPrisoners(allPrisoners);
        compositeAggregate.addAggregate(personDataProvider);
        compositeAggregate.addAggregate(prisonersDatabase);
    }

    public Finder(PersonDataProvider personDataProvider, PrisonersDatabase prisonersDatabase) {
        compositeAggregate.addAggregate(personDataProvider);
        compositeAggregate.addAggregate(prisonersDatabase);
    }
```

```java
    public void displayAllSuspectsWithName(String name) {
        ArrayList<Suspect> suspected = new ArrayList<Suspect>();

        for (Iterator<? extends Suspect> it = compositeAggregate.iterator(); it.hasNext(); ) {
            Suspect suspect = it.next();
            if (suspect.canBeAccused() && suspect.firstName().equals(name)) {
                suspected.add(suspect);
            }
            if (suspected.size() >= 10) {
                break;
            }
        }

        System.out.println("Znalazlem " + suspected.size() + " pasujacych podejrzanych!");

        for (Suspect n : suspected) {
            System.out.println(n.display());
        }

    }
}
```

W konstruktorze Findera nie powinny znajdować się konstruktor z kolekcjami, ja i konstruktor z PersonDataProvider, oraz PrisonerDatabase jednak bez tego wysypywały się testy i aplikacja. Ja bym to zrobił tak aby trzeba było dodać nowy SuspectAggreagate tylko do CompositeAggregate gdy chcemy takowy dodać.

Krok 6

Interfejs SearchStrategy

```java
package pl.edu.agh.to.lab4;

public interface SearchStrategy {
    boolean filter(Suspect suspect);
}
```

AgeSearchStrategy:

```java
package pl.edu.agh.to.lab4;

public class AgeSearchStrategy implements  SearchStrategy{

    private int age;

    public AgeSearchStrategy(int age) {
        this.age = age;
    }

    @Override
    public boolean filter(Suspect suspect) {
        return suspect.getAge() > age;
    }

}
```

NameSearchStrategy:

```java
package pl.edu.agh.to.lab4;

public class NameSearchStrategy implements SearchStrategy {

    private String name;

    public NameSearchStrategy(String name) {
        this.name = name;
    }

    @Override
    public boolean filter(Suspect suspect) {
        return suspect.firstName.equals(name);
    }
}
```

CompositeSearchStrategy:

```java
public class CompositeSearchStrategy implements SearchStrategy{

    Collection<SearchStrategy> strategies = new ArrayList<>();

    @Override
    public boolean filter(Suspect suspect) {
        boolean flag = true;
        for (SearchStrategy s : strategies){
            if(!s.filter(suspect)){
                flag = false;
                break;
            }
        }
        return flag;
    }

    public void addSearchStrategy(SearchStrategy s){
        strategies.add(s);
    }
}
```

Finder:

```java
public class Finder {

    private CompositeAggregate compositeAggregate;

    public Finder(CompositeAggregate compositeAggregate) {
        this.compositeAggregate = compositeAggregate;
    }

    public void display(SearchStrategy strategy) {
        ArrayList<Suspect> suspected = new ArrayList<Suspect>();

        for (Iterator<? extends Suspect> it = compositeAggregate.iterator(); it.hasNext(); ) {
            Suspect suspect = it.next();
            if (suspect.canBeAccused() && strategy.filter(suspect)) {
                suspected.add(suspect);
            }
            if (suspected.size() >= 10) {
                break;
            }
        }

        System.out.println("Znalazlem " + suspected.size() + " pasujacych podejrzanych!");

        for (Suspect n : suspected) {
            System.out.println(n.display());
        }

    }
}
```

Aplikacja:

```java
public class Application {

    public static void main(String[] args) {
        CompositeAggregate compositeAggregate = new CompositeAggregate();
        compositeAggregate.addAggregate(new PersonDataProvider());
        compositeAggregate.addAggregate(new PrisonersDatabase());

        Finder finder = new Finder(compositeAggregate);

        CompositeSearchStrategy strategy = new CompositeSearchStrategy();
        strategy.addSearchStrategy(new AgeSearchStrategy(18));
        strategy.addSearchStrategy(new NameSearchStrategy("Krzysztof"));

        finder.display(strategy);

    }
}
```

```
Znalazlem 1 pasujacych podejrzanych!
Krzysztof Mendel

Process finished with exit code 0
```

W bazie więźniów nie był wprowadzony wiek więc żeby móc przetestować szukanie po wieku dodałem do klasy abstrakcyjnej:

```
public abstract class Suspect {

    protected String firstName;
    protected String lastName;
    protected int Age = 20;
```

```
CompositeSearchStrategy strategy = new CompositeSearchStrategy();
strategy.addSearchStrategy(new AgeSearchStrategy(18));
strategy.addSearchStrategy(new NameSearchStrategy("Janusz"));

finder.display(strategy);
```

```
Znalazlem 5 pasujacych podejrzanych!
Janusz Krakowski
Janusz Programista
Janusz Zlowieszczy
Janusz Podejrzany
Janusz Zamkniety

Process finished with exit code 0
```

Dodajemy nowe baze podejrzanych

Student:

```java
package pl.edu.agh.to.lab4;

public class Student extends Suspect {

    private int index;

    public Student(String firstName, String lastName, int age,int index) {
        super(firstName, lastName, age);
        this.index = index;
    }

    @Override
    boolean canBeAccused() { return true; }

}
```

StudentDataProvider:

```java
public class StudentDataProvider implements SuspectAggreagate{

    private Collection<Student> students = new ArrayList<Student>();

    public StudentDataProvider(Collection<Student> students) {
        this.students = students;
    }

    public StudentDataProvider(){
        students.add(new Student( firstName: "Janusz", lastName: "Polak", age: 23, index: 22222));
        students.add(new Student( firstName: "Jan", lastName: "Polak", age: 23, index: 22222));
    }

    @Override
    public Iterator<? extends Suspect> iterator() { return students.iterator(); }
}
```

Aplikacja:

```java
public static void main(String[] args) {
  CompositeAggregate compositeAggregate = new CompositeAggregate();
  compositeAggregate.addAggregate(new PersonDataProvider());
  compositeAggregate.addAggregate(new PrisonersDatabase());
  compositeAggregate.addAggregate(new StudentDataProvider());
```

```java
Finder finder = new Finder(compositeAggregate);

CompositeSearchStrategy strategy = new CompositeSearchStrategy();
strategy.addSearchStrategy(new AgeSearchStrategy(18));
strategy.addSearchStrategy(new NameSearchStrategy("Janusz"));
```

```
Znalazlem 6 pasujacych podejrzanych!
Janusz Krakowski
Janusz Programista
Janusz Zlowieszczy
Janusz Podejrzany
Janusz Zamkniety
Janusz Polak
```

Widzimy naszego Studenta