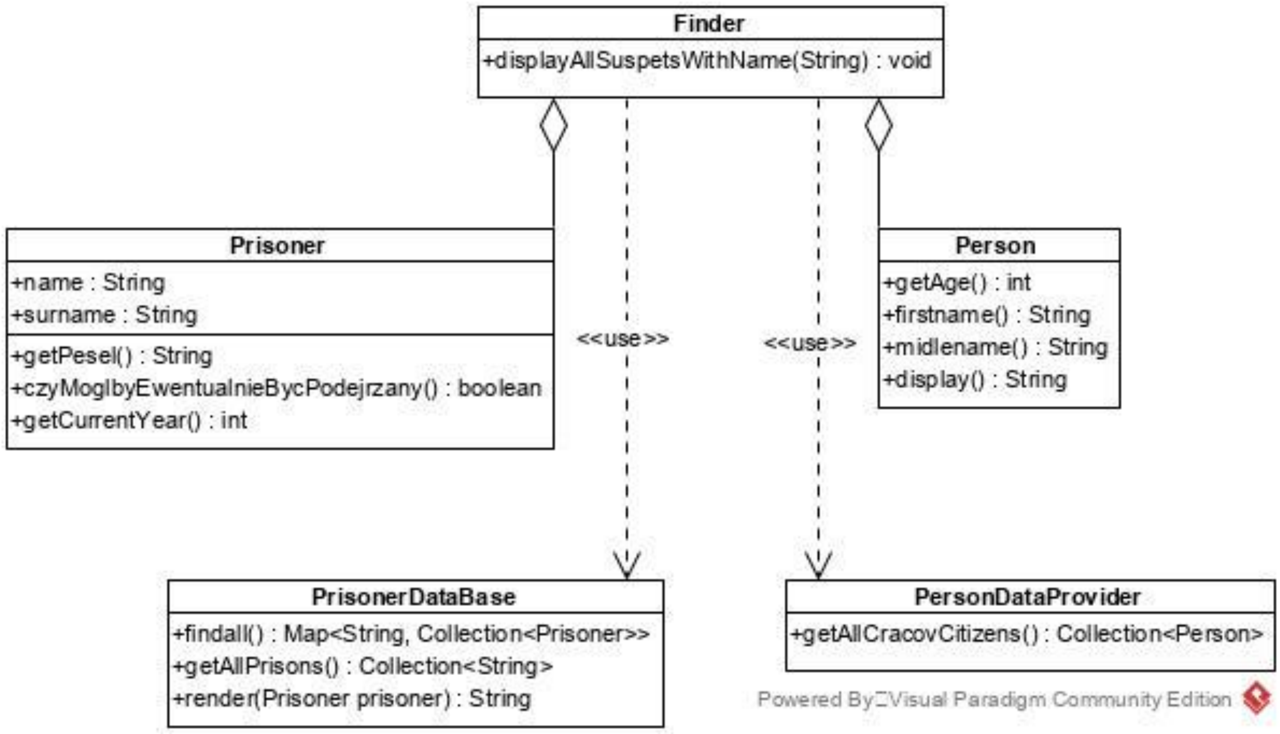
# Projektowanie obiektowe

Lab 4 Refaktoryzacja

Dawid Białka, Piotr Tekielak

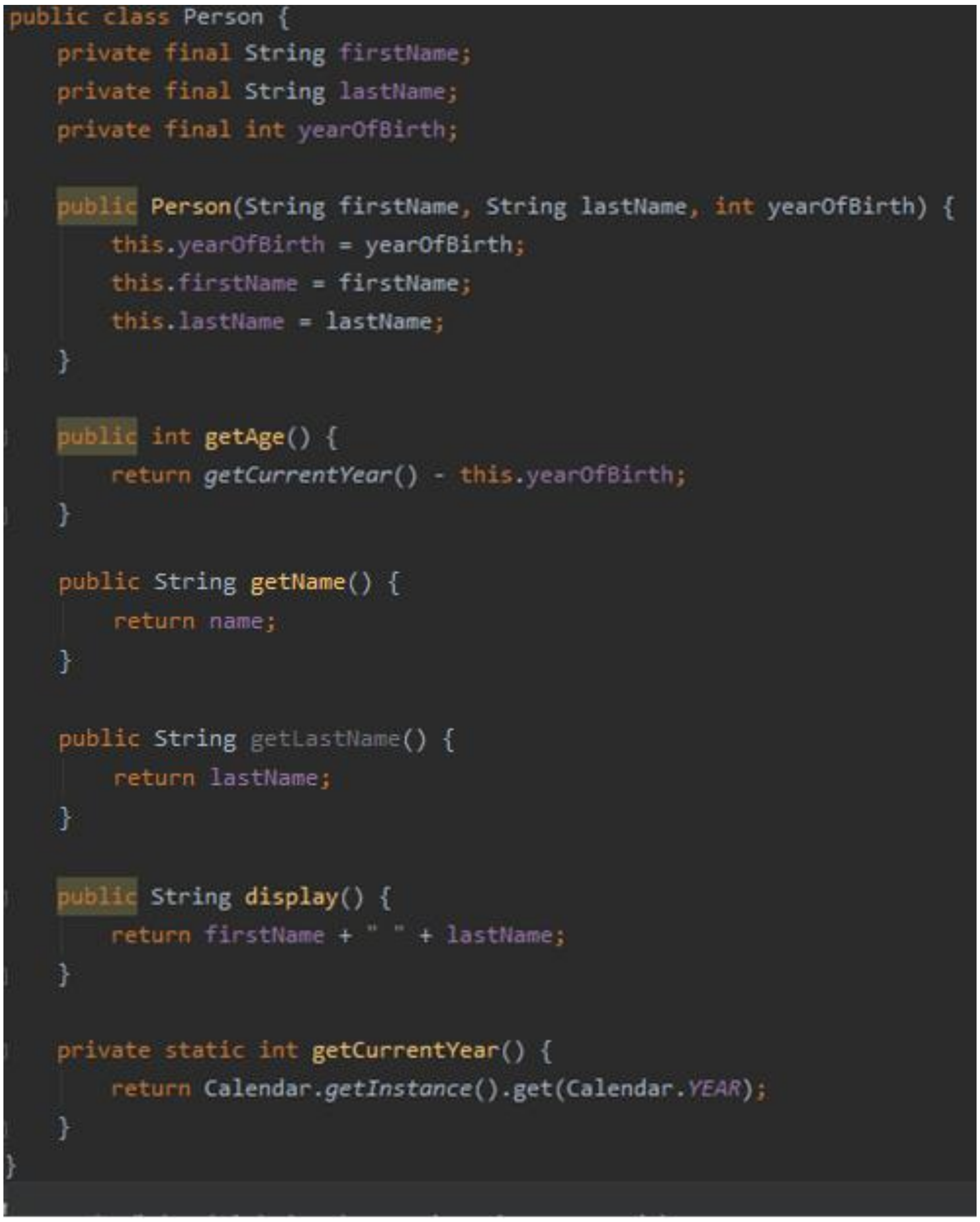
## 1. Diagram UML


Finder
+displayAllSuspetsWithName(String) : void
Prisoner
+name : String
+surname : String
+getPesel() : String
+czyMoglbyEwentualnieBycPodejrzany() : boolean
+getCurrentYear() : int
Person
+getAge() : int
+firstname() : String
+midlename() : String
+display() : String
<<use>>
<<use>>
PrisonerDataBase
+findall() : Map<String, Collection<Prisoner>>
+getAllPrisons() : Collection<String>
+render(Prisoner prisoner) : String
PersonDataProvider
+getAllCracovCitizens() : Collection<Person>
Powered By Visual Paradigm Community Edition

W podanym kodzie Finder tylko używa PrisonerDataBase i PersonDataProvider jako parametru w konstruktorze. Natomiast na diagramie UML w instrukcji Finder posiada pola o tych typach.

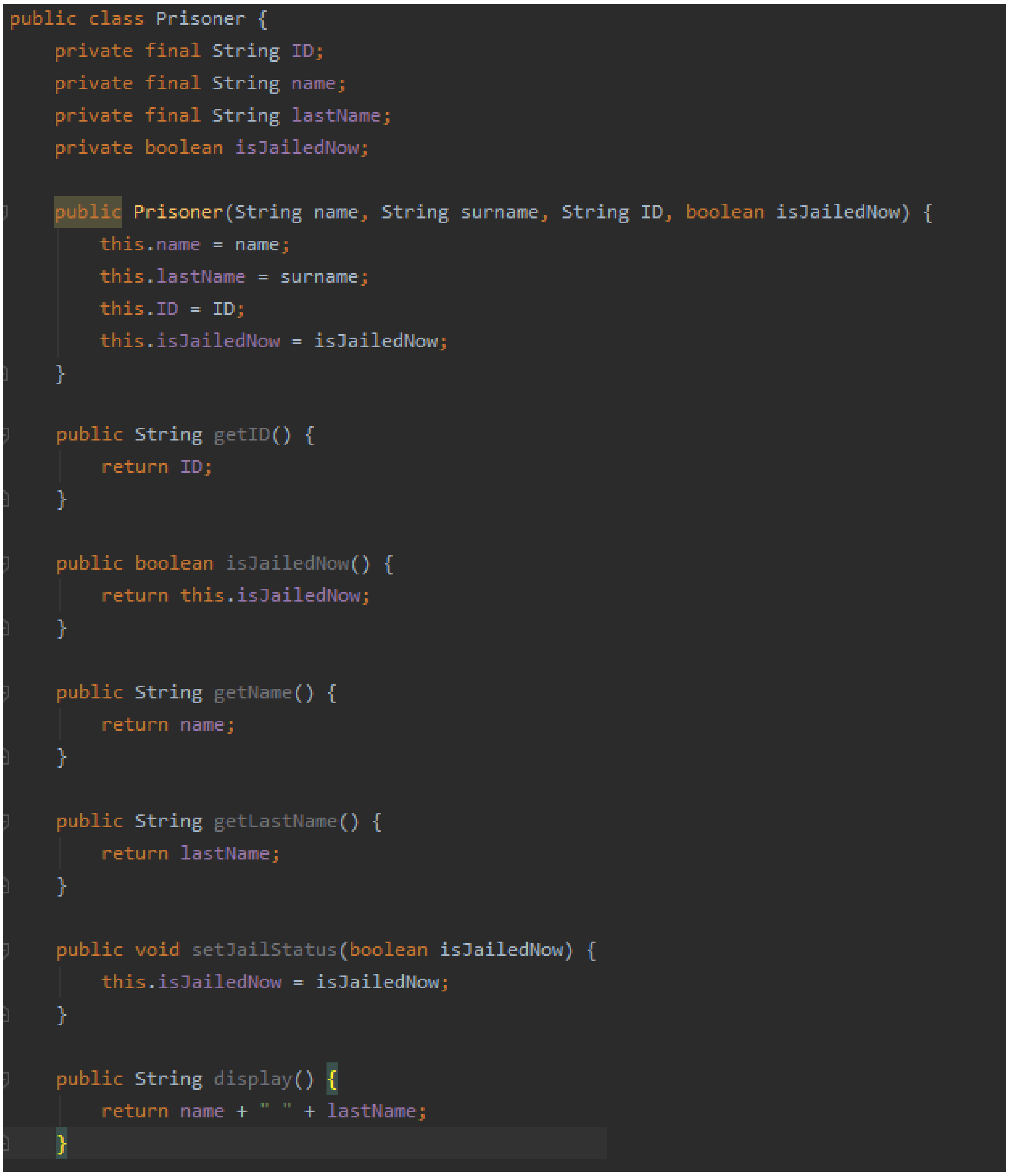
## 2. Poprawa podstawowych błędów.

Zmiana nazwy metod na getName i getLastName. Do pól name i lastName dodany final. Zamiast przechowywania wieku osoby, który trzeba aktualizować co rok, będziemy przechowywać rok urodzenia. Dodajemy prywatną, statyczną metodą getCurrentYear, która zwraca nam obecny rok i w metodzie getAge obliczamy wiek danej osoby.



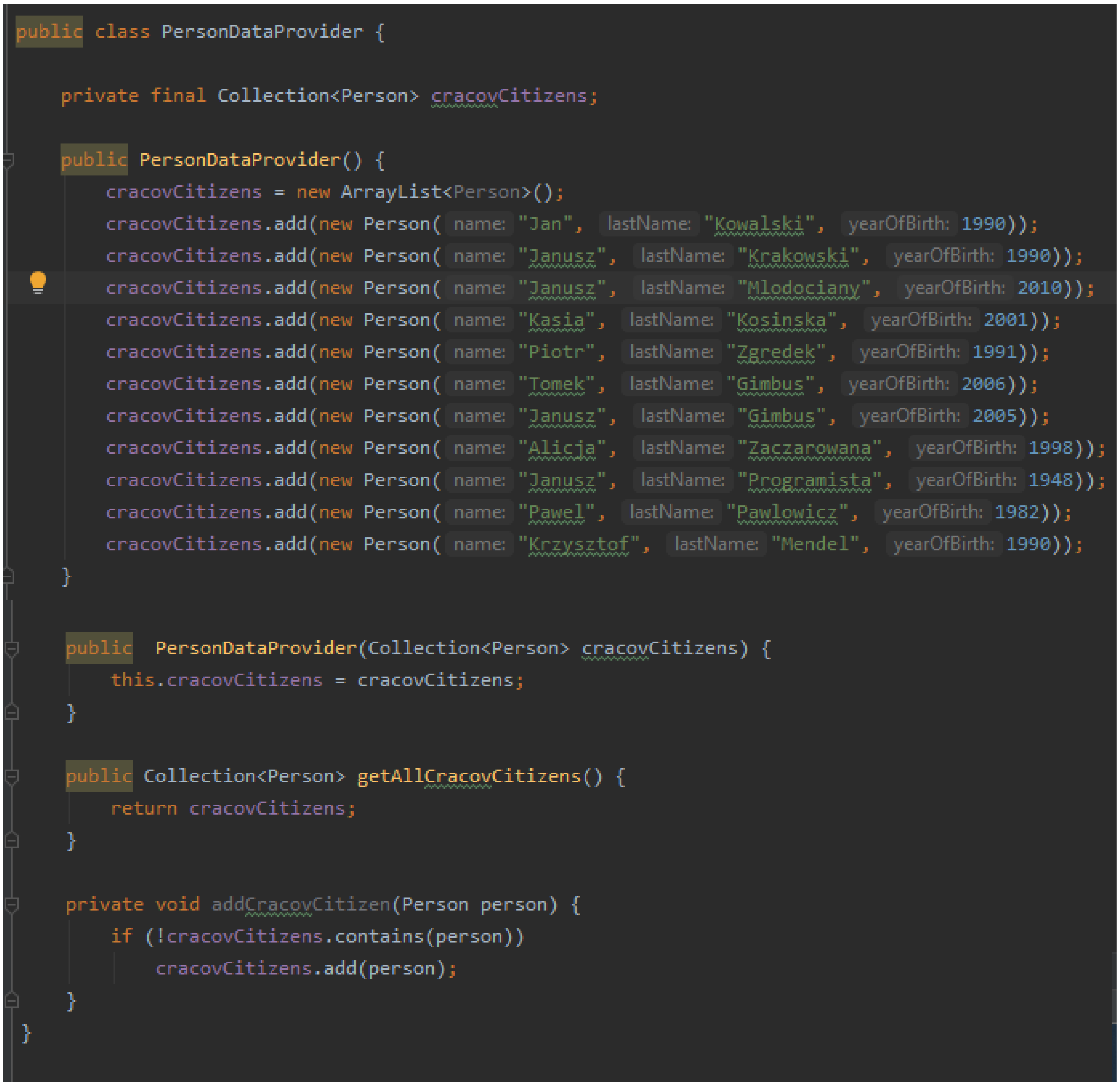
```
public class Person {
    private final String firstName;
    private final String lastName;
    private final int yearOfBirth;

    public Person(String firstName, String lastName, int yearOfBirth) {
        this.yearOfBirth = yearOfBirth;
        this.firstName = firstName;
        this.lastName = lastName;
    }

    public int getAge() {
        return getCurrentYear() - this.yearOfBirth;
    }

    public String getName() {
        return name;
    }

    public String getLastName() {
        return lastName;
    }

    public String display() {
        return firstName + " " + lastName;
    }

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

Zmiana nazwy pola pesel na ID. Wyrzucamy pola sentenceDuration i judgementYear zamieniając je na pole boolean isJailedNow. Zmiana nazwy metody czyMoglByBycEwentualniePodejrzany isJailedNow . Dodanie metody setJailStatus i display. Zmiana nazw metod na getName i getLastName.



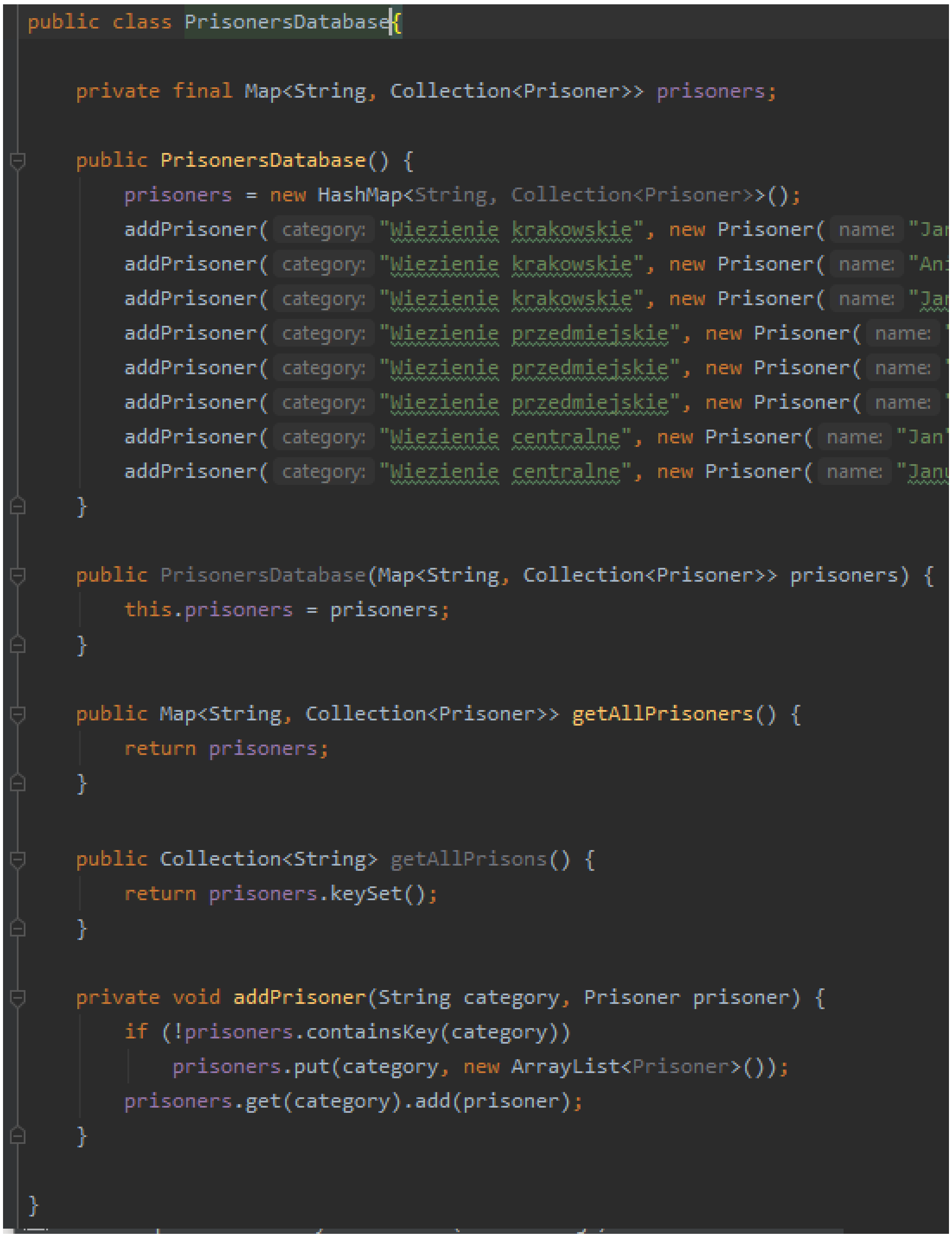
```java
public class Prisoner {
    private final String ID;
    private final String name;
    private final String lastName;
    private boolean isJailedNow;

    public Prisoner(String name, String surname, String ID, boolean isJailedNow) {
        this.name = name;
        this.lastName = surname;
        this.ID = ID;
        this.isJailedNow = isJailedNow;
    }

    public String getID() {
        return ID;
    }

    public boolean isJailedNow() {
        return this.isJailedNow;
    }

    public String getName() {
        return name;
    }

    public String getLastName() {
        return lastName;
    }

    public void setJailStatus(boolean isJailedNow) {
        this.isJailedNow = isJailedNow;
    }

    public String display() {
        return name + " " + lastName;
    }
```

Dodanie metody addCracovCitizen i konstruktora.



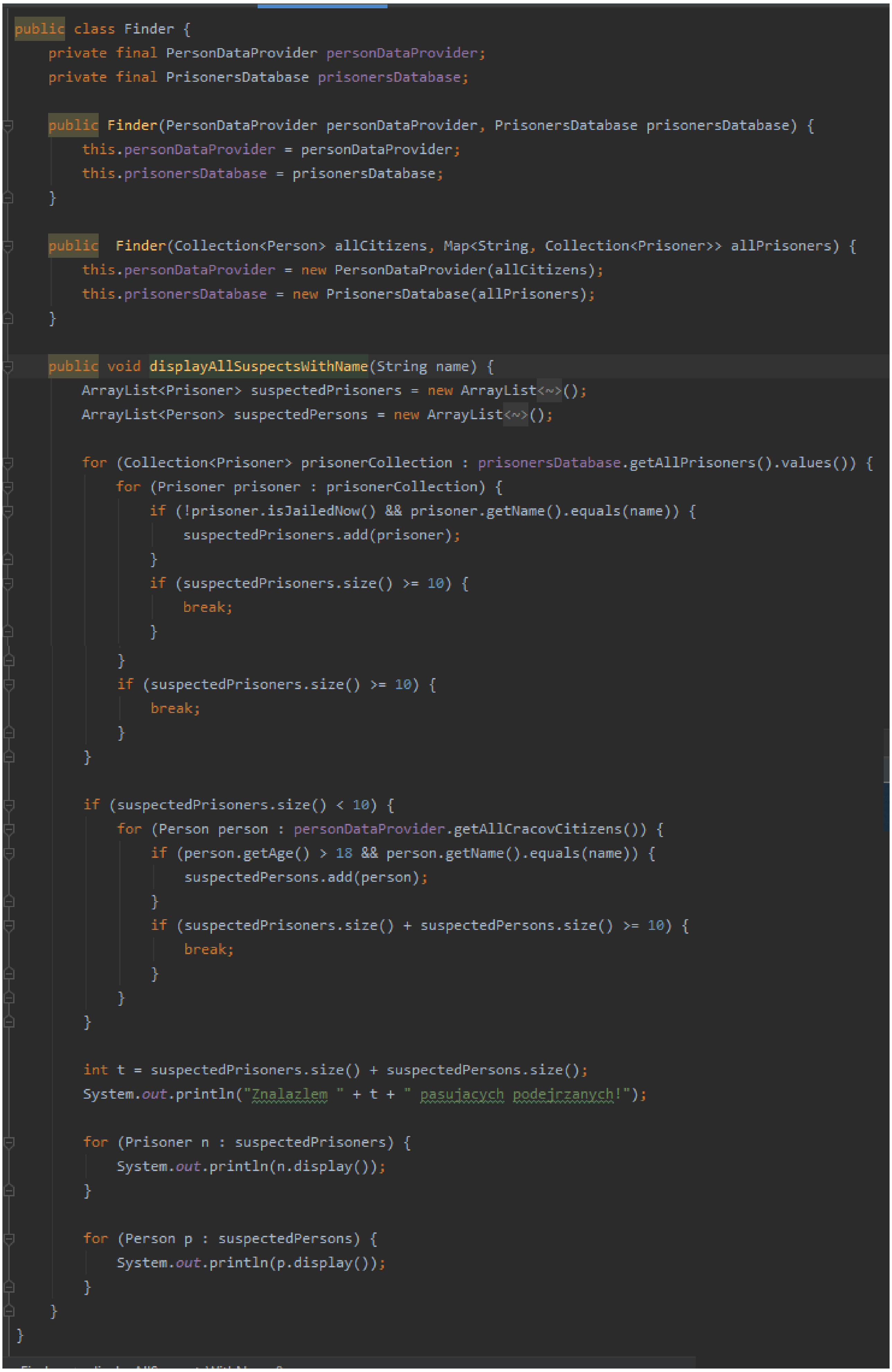
```java
public class PersonDataProvider {

    private final Collection<Person> cracovCitizens;

    public PersonDataProvider() {
        cracovCitizens = new ArrayList<Person>();
        cracovCitizens.add(new Person( name: "Jan", lastName: "Kowalski", yearOfBirth: 1990));
        cracovCitizens.add(new Person( name: "Janusz", lastName: "Krakowski", yearOfBirth: 1990));
        cracovCitizens.add(new Person( name: "Janusz", lastName: "Mlodociany", yearOfBirth: 2010));
        cracovCitizens.add(new Person( name: "Kasia", lastName: "Kosinska", yearOfBirth: 2001));
        cracovCitizens.add(new Person( name: "Piotr", lastName: "Zgredek", yearOfBirth: 1991));
        cracovCitizens.add(new Person( name: "Tomek", lastName: "Gimbus", yearOfBirth: 2006));
        cracovCitizens.add(new Person( name: "Janusz", lastName: "Gimbus", yearOfBirth: 2005));
        cracovCitizens.add(new Person( name: "Alicja", lastName: "Zaczarowana", yearOfBirth: 1998));
        cracovCitizens.add(new Person( name: "Janusz", lastName: "Programista", yearOfBirth: 1948));
        cracovCitizens.add(new Person( name: "Pawel", lastName: "Pawlowicz", yearOfBirth: 1982));
        cracovCitizens.add(new Person( name: "Krzysztof", lastName: "Mendel", yearOfBirth: 1990));
    }

    public  PersonDataProvider(Collection<Person> cracovCitizens) {
        this.cracovCitizens = cracovCitizens;
    }

    public Collection<Person> getAllCracovCitizens() {
        return cracovCitizens;
    }

    private void addCracovCitizen(Person person) {
        if (!cracovCitizens.contains(person))
            cracovCitizens.add(person);
    }
}
```

Usunięcie metody render i dodanie metody display do Prisoner.  Zmiana nazwy metody z findAll na getAllPrisoners. Dodanie nowego konstruktora.



```
public class PrisonersDatabase{

    private final Map<String, Collection<Prisoner>> prisoners;

    public PrisonersDatabase() {
        prisoners = new HashMap<String, Collection<Prisoner>>();
        addPrisoner( category: "Wiezienie krakowskie", new Prisoner( name: "Jar
        addPrisoner( category: "Wiezienie krakowskie", new Prisoner( name: "An:
        addPrisoner( category: "Wiezienie krakowskie", new Prisoner( name: "Jar
        addPrisoner( category: "Wiezienie przedmiejskie", new Prisoner( name: '
        addPrisoner( category: "Wiezienie przedmiejskie", new Prisoner( name: '
        addPrisoner( category: "Wiezienie przedmiejskie", new Prisoner( name: '
        addPrisoner( category: "Wiezienie centralne", new Prisoner( name: "Jan'
        addPrisoner( category: "Wiezienie centralne", new Prisoner( name: "Janu
    }

    public PrisonersDatabase(Map<String, Collection<Prisoner>> prisoners) {
        this.prisoners = prisoners;
    }

    public Map<String, Collection<Prisoner>> getAllPrisoners() {
        return prisoners;
    }

    public Collection<String> getAllPrisons() {
        return prisoners.keySet();
    }

    private void addPrisoner(String category, Prisoner prisoner) {
        if (!prisoners.containsKey(category))
            prisoners.put(category, new ArrayList<Prisoner>());
        prisoners.get(category).add(prisoner);
    }

}
```

Uaktualnienie wywołań metod na nowe nazwy.



```java
public class Finder {
    private final PersonDataProvider personDataProvider;
    private final PrisonersDatabase prisonersDatabase;

    public Finder(PersonDataProvider personDataProvider, PrisonersDatabase prisonersDatabase) {
        this.personDataProvider = personDataProvider;
        this.prisonersDatabase = prisonersDatabase;
    }

    public  Finder(Collection<Person> allCitizens, Map<String, Collection<Prisoner>> allPrisoners) {
        this.personDataProvider = new PersonDataProvider(allCitizens);
        this.prisonersDatabase = new PrisonersDatabase(allPrisoners);
    }

    public void displayAllSuspectsWithName(String name) {
        ArrayList<Prisoner> suspectedPrisoners = new ArrayList<~>();
        ArrayList<Person> suspectedPersons = new ArrayList<~>();

        for (Collection<Prisoner> prisonerCollection : prisonersDatabase.getAllPrisoners().values()) {
            for (Prisoner prisoner : prisonerCollection) {
                if (!prisoner.isJailedNow() && prisoner.getName().equals(name)) {
                    suspectedPrisoners.add(prisoner);
                }
                if (suspectedPrisoners.size() >= 10) {
                    break;
                }
            }
            if (suspectedPrisoners.size() >= 10) {
                break;
            }
        }

        if (suspectedPrisoners.size() < 10) {
            for (Person person : personDataProvider.getAllCracovCitizens()) {
                if (person.getAge() > 18 && person.getName().equals(name)) {
                    suspectedPersons.add(person);
                }
                if (suspectedPrisoners.size() + suspectedPersons.size() >= 10) {
                    break;
                }
            }
        }

        int t = suspectedPrisoners.size() + suspectedPersons.size();
        System.out.println("Znalazlem " + t + " pasujacych podejrzanych!");

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

        for (Person p : suspectedPersons) {
            System.out.println(p.display());
        }
    }
}
```

## 3. Generalizacja klas Person i Prisoner

Nowa klasa abstrakcyjna Suspect, po której dziedziczą Person i Prisoner.

```java
public abstract class Suspect {
    private final String name;
    private final String lastName;

    public Suspect(String name, String lastName) {
        this.name = name;
        this.lastName = lastName;
    }


    public String getName() {
        return name;
    }

    public String getLastName() {
        return lastName;
    }

    public String display() {
        return name + " " + lastName;
    }

    public abstract boolean canBeAccused();
}
```

Klasa Prisoner, dziedzicząca po Suspect. Przeniesienie metod getName, getLastName i display do Suspect.

```java
public class Prisoner extends Suspect {
    private final String ID;
    private boolean isJailedNow;

    public Prisoner(String name, String lastName, String ID, boolean isJailedNow) {
        super(name, lastName);
        this.ID = ID;
        this.isJailedNow = isJailedNow;
    }

    public String getID() {
        return ID;
    }

    public boolean isJailedNow() {
        return this.isJailedNow;
    }

    public void setJailStatus(boolean isJailedNow) {
        this.isJailedNow = isJailedNow;
    }

    public boolean canBeAccused () {
        return !this.isJailedNow();
    }

}
```

Klasa Person, dziedzicząca po Suspect. Przeniesienie metod getName, getLastName i display do Suspect.

```java
public class Person extends Suspect {
    private final int yearOfBirth;

    public Person(String name, String lastName, int yearOfBirth) {
        super(name, lastName);
        this.yearOfBirth = yearOfBirth;
    }

    public int getAge() {
        return getCurrentYear() - this.yearOfBirth;
    }

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

    public boolean canBeAccused () {
        return this.getAge() >= 18;
    }
}
```

Klasa Finder po wprowadzonych powyżej zmianach.

```java
public class Finder {
    private final SuspectAgregate personDataProvider;
    private final SuspectAgregate prisonersDatabase;

    public Finder(PersonDataProvider personDataProvider, PrisonersDatabase prisonersDatabase) {
        this.personDataProvider = personDataProvider;
        this.prisonersDatabase = prisonersDatabase;
    }

    public  Finder(Collection<Person> allCitizens, Map<String, Collection<Prisoner>> allPrisoners) {
        this.personDataProvider = new PersonDataProvider(allCitizens);
        this.prisonersDatabase = new PrisonersDatabase(allPrisoners);
    }


    public void displayAllSuspectsWithName(String name) {
        ArrayList<Suspect> suspects = new ArrayList<Suspect>();

        for (Collection<Prisoner> prisonerCollection : prisonersDatabase.getAllCracovCitizens()) {
            for (Prisoner prisoner : prisonerCollection) {
                if (prisoner.canBeAccused() && prisoner.getName().equals(name)) {
                    suspects.add(prisoner);
                }
                if (suspects.size() >= 10) {
                    break;
                }
            }
            if (suspects.size() >= 10) {
                break;
            }
        }

        if (suspects.size() < 10) {
            for (Person person : personDataProvider.getAllCracovCitizens()) {
                if (person.canBeAccused() && person.getName().equals(name)) {
                    suspects.add(person);
                }
                if (suspects.size() >= 10) {
                    break;
                }
            }
        }
        int t = suspects.size();
        System.out.println("Znalazlem " + t + " pasujacych podejrzanych!");

        for (Suspect s : suspects) {
            System.out.println(s.display());
        }
    }
}
```

## 4. Generalizacja klas PersonDataProvieder I PrisonerDataBase

Iterator dla PrisonerDataBase.

```
class PrisonersIterator implements Iterator<Suspect> {
    private ArrayList<Iterator<Prisoner>> suspects = new ArrayList<>();
    private int counter = 0;
    private int numOfIterators = 0;

    public PrisonersIterator(PrisonersDatabase prisonersDatabase)
    {
        for (Collection<Prisoner> prisonerCollection : prisonersDatabase.getAllPrisoners().values()) {
            suspects.add(prisonerCollection.iterator());
            numOfIterators++;
        }
    }

    @Override
    public boolean hasNext()
    {
        return suspects.get(numOfIterators-1).hasNext();
    }

    @Override
    public Suspect next()
    {
        if(!suspects.get(counter).hasNext()) {
            counter++;
        }
        return suspects.get(counter).next();
    }
}
```

Interfejs SuspectAgregate z metodą iterator.

```
public interface SuspectAgregate {
    Iterator<Suspect> iterator();
}
```

Klasa PrisonerDataBase z dodanym iteratorem i implementująca interfejs.

```java
public class PrisonersDatabase implements SuspectAgregate {

    private final Map<String, Collection<Prisoner>> prisoners;
    private final PrisonersIterator iterator;

    public PrisonersDatabase() {
        iterator = new PrisonersIterator(this);
        prisoners = new HashMap<String, Collection<Prisoner>>();
        addPrisoner("Wiezienie krakowskie", new Prisoner("Jan", "Kowalski", "87080452357",
                                                 false));
        addPrisoner("Wiezienie krakowskie", new Prisoner("Anita", "Wiercipieta", "84080452357",
                                                 false));
        addPrisoner("Wiezienie krakowskie", new Prisoner("Janusz", "Zlowieszczy",
                                                 "92080445657", false));
        addPrisoner("Wiezienie przedmiejskie", new Prisoner("Janusz", "Zamkniety",
                                                 "802104543357", false));
        addPrisoner("Wiezienie przedmiejskie", new Prisoner("Adam", "Future", "880216043357",
                                                 true));
        addPrisoner("Wiezienie przedmiejskie", new Prisoner("Zbigniew", "Nienajedzony",
                                                 "90051452335", false));
        addPrisoner("Wiezienie centralne", new Prisoner("Jan", "Przedziwny", "91103145223",
                                                 false));
        addPrisoner("Wiezienie centralne", new Prisoner("Janusz", "Podejrzany", "85121212456",
                                                 false));
    }

    public PrisonersDatabase(Map<String, Collection<Prisoner>> prisoners) {
        iterator = new PrisonersIterator(this);
        this.prisoners = prisoners;
    }

    public Iterator<Suspect> iterator() {
        return this.iterator;
    }

    public Map<String, Collection<Prisoner>> getAllPrisoners() {
        return prisoners;
    }

    public Collection<String> getAllPrisons() {
        return prisoners.keySet();
    }

    private void addPrisoner(String category, Prisoner prisoner) {
        if (!prisoners.containsKey(category))
            prisoners.put(category, new ArrayList<Prisoner>());
        prisoners.get(category).add(prisoner);
    }

}
```

Klasa PersonDataProvider z dodanym iteratorem i implementująca interfejs.

```java
public class PersonDataProvider implements SuspectAgregate{

    private final Collection<Person> cracovCitizens;

    public PersonDataProvider() {
        cracovCitizens = new ArrayList<Person>();
        cracovCitizens.add(new Person("Jan", "Kowalski", 1990));
        cracovCitizens.add(new Person("Janusz", "Krakowski", 1990));
        cracovCitizens.add(new Person("Janusz", "Mlodociany", 2010));
        cracovCitizens.add(new Person("Kasia", "Kosinska", 2001));
        cracovCitizens.add(new Person("Piotr", "Zgredek", 1991));
        cracovCitizens.add(new Person("Tomek", "Gimbus", 2006));
        cracovCitizens.add(new Person("Janusz", "Gimbus", 2005));
        cracovCitizens.add(new Person("Alicja", "Zaczarowana", 1998));
        cracovCitizens.add(new Person("Janusz", "Programista", 1948));
        cracovCitizens.add(new Person("Pawel", "Pawlowicz", 1982));
        cracovCitizens.add(new Person("Krzysztof", "Mendel", 1990));
    }

    public  PersonDataProvider(Collection<Person> cracovCitizens) {
        this.cracovCitizens = cracovCitizens;
    }

    public Iterator<Suspect> iterator() {
        ArrayList<Suspect> suspectArrayList = new ArrayList<Suspect>(cracovCitizens);
        return suspectArrayList.iterator();
    }

    public Collection<Person> getAllCracovCitizens() {
        return cracovCitizens;
    }

    private void addCracovCitizen(Person person) {
        if (!cracovCitizens.contains(person))
            cracovCitizens.add(person);
    }
}
```

Klasa Finder, która teraz posiada pola typu SuspectAgregate.

```java
public class Finder {
    private final SuspectAgregate personDataProvider;
    private final SuspectAgregate prisonersDatabase;

    public Finder(PersonDataProvider personDataProvider, PrisonersDatabase prisonersDatabase) {
        this.personDataProvider = personDataProvider;
        this.prisonersDatabase = prisonersDatabase;
    }

    public  Finder(Collection<Person> allCitizens, Map<String, Collection<Prisoner>> allPrisoners) {
        this.personDataProvider = new PersonDataProvider(allCitizens);
        this.prisonersDatabase = new PrisonersDatabase(allPrisoners);
    }

    public void displayAllSuspectsWithName(String name) {
        ArrayList<Suspect> suspects = new ArrayList<Suspect>();

        Iterator<Suspect> prisonerIterator = prisonersDatabase.iterator();
        Iterator<Suspect> personIterator = personDataProvider.iterator();

        while(prisonerIterator.hasNext() && suspects.size() < 10) {
           Prisoner currentPrisoner = (Prisoner) prisonerIterator.next();
           if (currentPrisoner.canBeAccused() && currentPrisoner.getName().equals(name)) {
               suspects.add(currentPrisoner);
           }
        }

        while(personIterator.hasNext() && suspects.size() < 10) {
            Person currentPerson = (Person) personIterator.next();
            if (currentPerson.canBeAccused() && currentPerson.getName().equals(name)) {
                suspects.add(currentPerson);
            }
        }

        int t = suspects.size();
        System.out.println("Znalazlem " + t + " pasujacych podejrzanych!");

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

## 5. Klasa pośrednia pomiędzy klasą Finder i SuspectAgregate.

Klasa CompositeAgregate wykorzystująca wzorzec Composite. Posiada tablicę  elementow SuspectAgregate i Iterator, dzięki któremu możemy przejść po wszystkich podejrzanych z wszystkich elementów SuspectAgregate.

```java
public class CompositeAgregate implements SuspectAgregate {
    private ArrayList<SuspectAgregate> childSuspectAgregate = new ArrayList<>();
    private Iterator<Suspect> iterator;

    public CompositeAgregate(Collection<SuspectAgregate> suspectAgregates) {
        childSuspectAgregate.addAll(suspectAgregates);
        ArrayList<Suspect> suspects = new ArrayList<>();

        for(SuspectAgregate agregate: childSuspectAgregate) {
            Iterator<Suspect> suspectIterator = agregate.iterator();

            while(suspectIterator.hasNext()) {
                suspects.add(suspectIterator.next());
            }
        }
        this.iterator = suspects.iterator();
    }

    public Iterator<Suspect> iterator() {
        return this.iterator;
    }
}
```

Klasa Finder posiadająca tylko pole compositeAgregate. Z tego pola pobiera iterator i przechodzi po wszystkich podejrzanych.

```java
public class Finder {
    private final CompositeAgregate compositeAgregate;

    public Finder(CompositeAgregate compositeAgregate) {
        this.compositeAgregate = compositeAgregate;
    }

    public void displayAllSuspectsWithName(String name) {
        ArrayList<Suspect> suspects = new ArrayList<Suspect>();

        Iterator<Suspect> suspectIterator = compositeAgregate.iterator();

        while(suspectIterator.hasNext() && suspects.size() < 10) {
           Suspect currentSuspect = suspectIterator.next();
            if (currentSuspect.canBeAccused() && currentSuspect.getName().equals(name)) {
                suspects.add(currentSuspect);
            }
        }

        int t = suspects.size();
        System.out.println("Znalazlem " + t + " pasujacych podejrzanych!");

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

Nowa wersja klasy Application.

```java
public class Application {

    public static void main(String[] args) {
        ArrayList<SuspectAgregate> suspectAgregates = new ArrayList<>();
        suspectAgregates.add(new PersonDataProvider());
        suspectAgregates.add(new PrisonersDatabase());
        Finder suspects = new Finder(new CompositeAgregate(suspectAgregates));
        suspects.displayAllSuspectsWithName("Janusz");
    }
}
```

## 6. SearchStrategy i końcowa wersja aplikacji.

Interfejs SearchStrategy.

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

Klasa NameSearchStrategy implementująca interfejs SearchStrategy.

```java
public class NameSearchStrategy implements SearchStrategy{
    private String name;

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

    public boolean filter(Suspect suspect) {
        return suspect.getName().equals(this.name) && suspect.canBeAccused();
    }
}
```

Klasa AgeSearchStrategy implementująca interfejs SearchStrategy.

```java
public class AgeSearchStrategy implements SearchStrategy{
    private int age;

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

    public boolean filter(Suspect suspect) {
        return suspect.getAge() == this.age && suspect.canBeAccused();
    }
}
```

Klasa CompositeSearchStrategy implementująca interfejs SearchStrategy i wykorzystująca wzorzec Composite. Filtrowane są osoby, które spełniają wszystkie warunki w liście strategii wyszukiawnia.

```java
public class CompositeSearchStrategy implements SearchStrategy{
    private ArrayList<SearchStrategy> childSearchStrategy = new ArrayList<>();

    public CompositeSearchStrategy(Collection<SearchStrategy> searchStrategies) {
        this.childSearchStrategy.addAll(searchStrategies);
    }

    public boolean filter(Suspect suspect) {
        for(SearchStrategy searchStrategy: childSearchStrategy) {
            if(!searchStrategy.filter(suspect)) {
                return false;
            }
        }
        return true;
    }
}
```

Klasa Student dziedzicząca po klasie Suspect.

```java
public class Student extends Suspect {
    private final int index;

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

    public boolean canBeAccused () {
        return this.getAge() >= 18;
    }
}
```

Klasa StudentDatabase przechowująca studentów z Krakowa i implementująca interfejs SuspectAgregate.

```java
public class StudentDatabase implements SuspectAgregate{

    private final Collection<Student> cracovStudents;

    public StudentDatabase() {
        cracovStudents = new ArrayList<>();
        cracovStudents.add(new Student("Zbyszko", "Zbogdanca", 1990, 345678));
        cracovStudents.add(new Student("Jurand", "Zespychowa", 1980, 234567));
        cracovStudents.add(new Student("Wladyslaw", "Jagiello", 2010, 654321));
        cracovStudents.add(new Student("Janusz", "Student", 1990, 123456));
    }

    public  StudentDatabase(Collection<Student> cracovStudents) {
        this.cracovStudents = cracovStudents;
    }

    public Iterator<Suspect> iterator() {
        ArrayList<Suspect> suspectArrayList = new ArrayList<Suspect>(cracovStudents);
        return suspectArrayList.iterator();
    }

    public Collection<Student> getAllCracovStudents() {
        return cracovStudents;
    }

    private void addCracovStudent(Student student) {
        if (!cracovStudents.contains(student))
            cracovStudents.add(student);
    }
}
```

Ostateczna wersja klasy Finder.

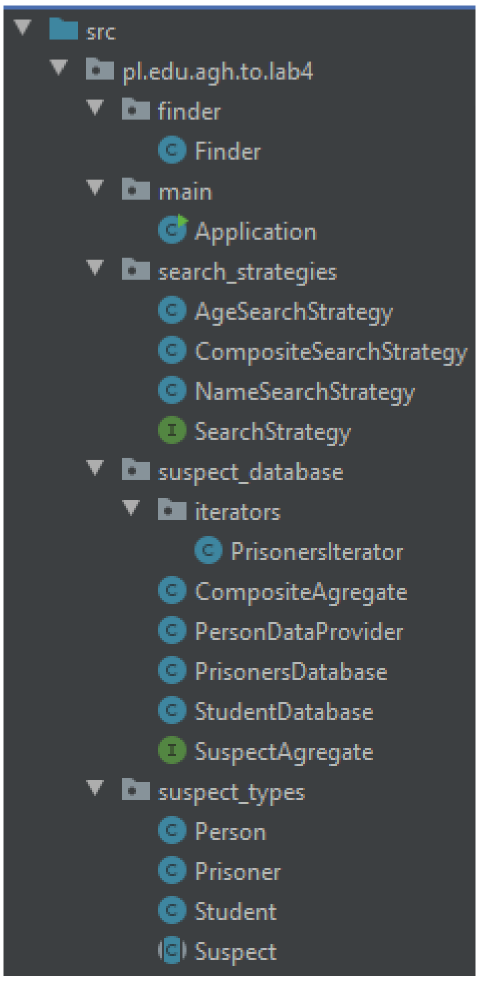
```java
public class Finder {
    private final CompositeAgregate compositeAgregate;
    private final SearchStrategy searchStrategy;

    public Finder(CompositeAgregate compositeAgregate, SearchStrategy searchStrategy) {
        this.compositeAgregate = compositeAgregate;
        this.searchStrategy = searchStrategy;
    }
    public void displayAllSuspectsWithName(String name) {
        ArrayList<Suspect> suspects = new ArrayList<Suspect>();

        Iterator<Suspect> suspectIterator = compositeAgregate.iterator();

        while(suspectIterator.hasNext() && suspects.size() < 10) {
           Suspect currentSuspect = suspectIterator.next();
            if (currentSuspect.canBeAccused() && currentSuspect.getName().equals(name)) {
                suspects.add(currentSuspect);
            }
        }

        int t = suspects.size();
        System.out.println("Znalazlem " + t + " pasujacych podejrzanych!");

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

Ostateczna wersja klasy Application.

```java
public class Application {

    public static void main(String[] args) {
        ArrayList<SuspectAgregate> suspectAgregates = new ArrayList<>();
        ArrayList<SearchStrategy> searchStrategies = new ArrayList<>();
        suspectAgregates.add(new PersonDataProvider());
        suspectAgregates.add(new PrisonersDatabase());
        suspectAgregates.add(new StudentDatabase());

        searchStrategies.add(new NameSearchStrategy("Janusz"));
        searchStrategies.add(new AgeSearchStrategy(30));

        Finder finder = new Finder(new CompositeAgregate(suspectAgregates), new
CompositeSearchStrategy(searchStrategies));
        finder.displayAllSuspectsWithName("Janusz");
    }
}
```

Reszta klas nie uległa zmianie od ich najnowszej wersji w powyższych podpunktach.

Na koniec zostały również dodane pakiety, aby posegregować odpowiednio klasy.



```
src
  pl.edu.agh.to.lab4
    finder
      Finder
    main
      Application
    search_strategies
      AgeSearchStrategy
      CompositeSearchStrategy
      NameSearchStrategy
      SearchStrategy
    suspect_database
      iterators
        PrisonersIterator
      CompositeAgregate
      PersonDataProvider
      PrisonersDatabase
      StudentDatabase
      SuspectAgregate
    suspect_types
      Person
      Prisoner
      Student
      Suspect
```