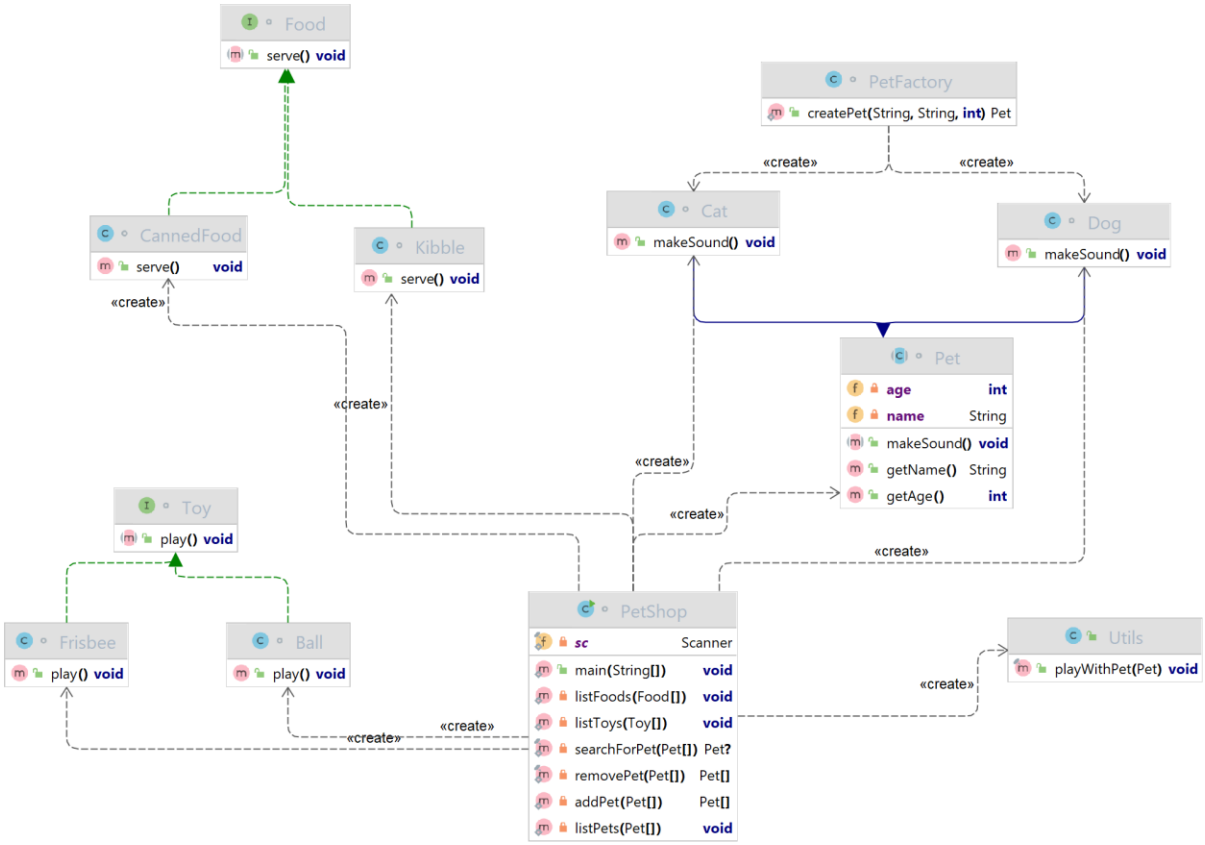
BIP2007 OOP-II FINAL EXAM PROJECT REPORT

In this project I implemented a Pet Shop project. Which has pets, foods, and toys. I tried to use most of the topics we cover in lectures and followed the rules given in the project document. My project has clear inheritance hierarchy including abstract classes, interfaces, and regular classes. I implement two different shapes of pets, foods and toys. In the main method, I create list of them and add functionality that user can reaches existing pets, adds new ones, or deletes one of them.



UML Diagram

I use final keyword for a method in Utils class. Because I will not override the method “playWithPet()”. We use final keyword for methods that will not be overridden by its super classes.

```
public class Utils {
    public final void playWithPet(Pet pet) {
        System.out.println("Playing with " + pet.getName() + "...");
        pet.makeSound();
    }
}
```

Example of using final keyword in a method

I also use final keyword in Pet class. Because I don't want method "getName()" and "getAge()" to be overridden.

```
public final String getName() {
    return name;
}

public final int getAge() {
    return age;
}
```

Another example of using final keyword in a method

I use static keyword for a variable in Pet class. It has variable named numberOfPets which is static int and takes how many pets created until now. Because of the pet removal functionality, I added to project, I also need to decrease it in some cases. Static variables are variables we use for the classes. Created objects does not have that variable. But the class, the object created from, has.

```
public static int numberOfPets;
```

Usage of static keyword in a variable

I used factory pattern as design pattern. I used it to create Pet objects easily. I declared a PetFactory class and created an instance of it in main method. When I need to create a new Pet, I just call that classes function.

```
class PetFactory {
    public static Pet createPet(String type, String name, int age) {
        switch (type) {
            case "dog":
                return new Dog(name, age);
            case "cat":
                return new Cat(name, age);
            default:
                throw new IllegalArgumentException("Invalid pet type: " + type);
        }
    }
}
```

Usage of factory pattern