PROGRAMOWANIE W JAVA

Laboratorium 2

Poniżej znajdują się zadania, które zostaną przerobione na drugich laboratoriach z przedmiotu “Programowanie w Java” studiów podyplomowych Programowanie I Bazy Danych.

Zadanie 1

Wygeneruj projekt Maven’owy o następującej konfiguracji:

- groupId: pl.edu.pg.ftims.itj
- artifactId: lab2
- versión: 1.0

Zadanie 2

Mając poniższą klasę Animal:

```
package pl.edu.pg.ftims.itj.animals;

public class Animal {
	public void sleep() {
		System.out.println("An animal sleeps...");
	}

	public void eat() {
		System.out.println("An animal eats...");
	}
}
```

1. Stwórz dwie nowe klasy – Bird oraz Dog. Obie powinny dziedziczyć po klasie Animal oraz nadpisywać metody sleep() oraz eat().
2. Stwórz klasę AnimalsInheritance z metodą main, w której stworzone zostaną obiekty klas Animal, Bird oraz Dog i wywołane odpowiednio dla każdego obiektu obie metody.

3. Stwórz klasę Eagle, która dziedziczy po klasie Bird i nadpisuje metody eat() oraz sleep(). Dodaj utworzenie obiektu klasy Eagle w klasie AnimalsInheritance i wywołaj nadpisane metody.
4. W implementacji metody sleep() w klasach Bird, Dog oraz Eagle dodaj wywołanie metody sleep() z klasy przodka. Przetestuj zmianę uruchamiając klasę Animals Inheritance.
5. Do klasy Animal dodaj statyczną metodę sleepAnimal(Animal animal), która będzie wywoływać metodę sleep() przekazanego obiektu. W metodzie main klasy AnimalsInheritance dodaj wywołanie metody sleepAnimal(…) z różnymi parametrami. Co obserwujesz?
6. W klasie AnimalsInheritance dodaj dla każdego stworzonego obiektu wywołanie metody getClass().getName() i wyświetl rezultat na ekranie. Co obserwujesz?
7. W klasie AnimalsInheritance dodaj dla każdego obiektu sprawdzenie, czy obiekt jest typu Animal, Dog, Bird oraz Eagle i wynik wypisz na ekranie. Co obserwujesz?

Zadanie 3

W klasie AnimalsInheritance dodaj dla każdego stworzonego obiektu wywołanie metody sleep() oraz eat() przy użyciu rzutowania na typ Animal. Jakiego typu rzutowanie to jest? Co ono oznacza? Co zostało wyświetlone na ekranie i dlaczego?

Zadanie 4

Mając poniższą klasę Animal:

```
package pl.edu.pg.ftims.itj.noises;

public class Animal {
    private String species;

        public Animal(String species) {
            this.species = species;
        }

        public void makeNoise() {
            if (species.equals("cat")) {
                System.out.println("Miau Miau");
            } else if (species.equals("mouse")) {
                System.out.println("Fiep Fiep");
            }
        }
}
```

1. Stwórz klasę AnimalTest z metodą main, w której stworzone zostaną dwa obiekty (z parametrami konstruktora odpowiednio “cat” oraz “mouse”) i wywołana zostanie metoda makeNoise.
2. Przed uruchomieniem programu na kartce zapisz jaki będzie według Ciebie rezultat uruchomienia klasy AnimalTest.
3. Uruchom klasę AnimalTest i zweryfikuj swoją odpowiedź na kartce z komunikatem na konsoli.
4. Zmień klasę Animal na abstrakcyjną z abstrakcyjną metodą makeNoise.
5. Stwórz dwie klasy Cat oraz Mouse dziedziczące po klasie Animal i implementujące metodę makeNoise()
6. W klasie AnimalTest stwórz obiekty klas Cat i Mouse oraz wywołaj ich metody makeNose() – wynik działania programu powinien pozostać bez zmian.

Zadanie 5

1. Stwórz pakiet pl.edu.pg.ftims.itj.overriding
2. W nowym pakiecie utwórz następujące klasy:
    a. public class Animal {}
    b. public class Fish extends Animal {}
    c. public class Bird extends Animal {}
    d. public class Chicken extends Bird {}
    e. public class Eagle extends Bird {}
    f. public class Karp extends Fish {}
3. Stwórz nową klasę Super o następującej implementacji:

```
public class Super {
	public void m(Animal a1, Animal a2) {
		System.out.println("1");
	}

	public void m(Animal a, Fish f) {
		System.out.println("2");
	}

	public void m(Fish f, Animal a) {
		System.out.println("5");
	}
}
```

4. Stwórz klasę Sub o następującej implementacji:

```
public class Sub extends Super {
	public void m(Animal a, Fish f) {
		System.out.println("3");
	}

	public void m(Bird b, Fish f) {
		System.out.println("4");
	}
}
```

5. Stwórz klasę Test z metodą main o następującej implementacji:

```
Animal a1 = new Animal();
Animal a2 = new Bird();
Fish f = newKarp();
Bird b1 = new Bird();
Bird b2 = new Eagle();
Chicken c = new Chicken();
Karp k = new Karp();

Super sup1 = newSuper();
Super sup2 = newSub();

sup1.m(c, b2); // call #1
sup2.m(b1, k); // call #2
sup1.m(a1, a2); // call #3
sup1.m(b1, k); // call #4
sup1.m(b2, f); // call #5
sup2.m(b1, f); // call #6
```

6. Przed uruchomieniem klasy Test zapisz na kartce jaki wynik będzie miało każde wywołanie metody m(…)
7. Uruchom klasę Test i zweryfikuj swoje zapiski

Zadanie 6

1. Stwórz pakiet pl.edu.pg.ftims.itj.singleton
2. Stwórz klasę SimpleSingleton, która posiada prywatny konstruktor
3. Wraz z prowadzącym zaimplementuj w klasie SimpleSingleton wzorzec projektowy Singleton
4. Stwórz klasę SingletonTest, w której przetestujesz poprawność implementacji