# Exercises: Inheritance

This document defines the exercises for the ["Java Advanced" course @ Software University](https://softuni.bg/trainings/4375/java-oop-february-2024). Please submit your solutions (source code) to all below-described problems in [Judge](https://judge.softuni.bg/Contests/1580/Inheritance-Exercises).

## Person

NOTE: You need a public class Main. Create a package person.

You are asked to model an application for storing data about people. You should be able to have a Person and a Child. The child derives from the person. Every person has a **name** and an **age**. Your task is to model the application.

The **Person** class also has getters for the fields.

Create a **Child** class that inherits **Person** and has the same public constructor definition. However, do not copy the code from the **Person** class - **reuse the Person class's constructor**.

|  |
| --- |
| **Sample** Main() |
| **public class** Main {  **public static void** main(String[] args) {  Scanner sc = **new** Scanner(System.***in***);   String name = sc.nextLine();  **int** age = Integer.*parseInt*(sc.nextLine());   Child child = **new** Child(name, age);   System.***out***.println(child.getName());  System.***out***.println(child.getAge());  } } |

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Peter  13 | Peter  13 |
| George  10 | George  10 |

## Zoo

**NOTE**: You need a public class Main.

Create a package zoo. It needs to contain the following classes:



Follow the diagram and create all of the classes. **Each** of them, except the **Animal** class, should **inherit** from **another** **class**. The Animal class should have a field name – String and **Getter** for a name.

Every class should have:

* A public constructor, which accepts one parameter: **name**

Zip your package and upload it to Judge.

## Players and Monsters

NOTE: You need a public class **Main**. Create a package hero.

Your task is to create the following game hierarchy:



Create a class **Hero**. It should contain the following members:

* A public constructor, which accepts:
  + **username – String**
  + **level – int**
* The following fields:
  + **username - String**
  + **level – int**
* Getters for username and level
* **toString()** method

Hint: Override **toString()** of the base class in the following way:

|  |
| --- |
| **Sample toString()** |
| @Override **public** String toString() {  **return** String.*format*(**"Type: %s Username: %s Level: %s"**,  **this**.getClass().getName(),  **this**.getUsername(),  **this**.getLevel()); } |

## Need for Speed

NOTE: You need a public class **Main**. Create the following **hierarchy** with the following **classes**: 

Create a base class Vehicle. It should contain the following members:

* **DEFAULT\_FUEL\_CONSUMPTION – final static double (constant)**
* **fuelConsumption – double**
* **fuel – double**
* **horsePower – int**
* **Getters and Setters for the fields**
* A public constructor which accepts (fuel, horsePower) and **set** the **default fuel consumption** on the field fuelConsumption
* **void drive(double kilometers)**
  + The **drive** method should have the functionality to reduce the **fuel** based on the traveled kilometers and fuel consumption. Keep in mind that you can drive the vehicle only if you have enough fuel to finish the driving.

The default fuel consumption for **Vehicle** is 1.25.Some of the classes have different default fuel consumption:

* **SportCar – DEFAULT\_FUEL\_CONSUMPTION = 10**
* **RaceMotorcycle – DEFAULT\_FUEL\_CONSUMPTION = 8**
* **Car – DEFAULT\_FUEL\_CONSUMPTION = 3**

Zip your package and upload it to Judge.

### Hint

In the child classes' constructors use super.setFuelConsumption() to set fuelConsumption.

## Restaurant

NOTE: You need a public class **Main**. Create a **restaurant** package with the following classes and hierarchy:

There are **Food** and **Beverages** in the restaurant and they are all products.

The **Product** class must have the following members:

* A public constructor with the following parameters: **String name, BigDecimal price**
* **name – String**
* **price – BigDecimal**
* **Getters for the fields**

**Beverage** and **Food** classes are products. The **Beverage** class must have the following members:

* A public constructor with the following parameters: **String name, BigDecimal price, double milliliters**
* **name – String**
* **price – BigDecimal**
* **milliliters - double**
* **Getter for milliliters**

The Food class must have the following members:

* A constructor with the following parameters: **String name, BigDecimal price, double grams**
* **name – String**
* **price – double**
* **grams - double**
* **Getter for grams**

**HotBeverage** and **ColdBeverage** are **beverages** and they accept the following parameters upon initialization: **String name, BigDecimal price, double milliliters**

**Coffee** and **Tea** are hot beverages. The **Coffee** class must have the following additional members:

* **double COFFEE\_MILLILITERS = 50**
* **BigDecimal COFFEE\_PRICE = 3.50**
* **caffeine – double**
* **Getter for caffeine**

MainDish, Dessert, and Starter are food. They all accept the following parameters upon initialization: **String name, BigDecimal price, double grams**. Dessert must accept one more parameter in its constructor: **double calories**.

* **calories – double**
* **Getter for calories**

Make **Salmon**, **Soup** and **Cake** inherit the proper classes.

A **Cake** must have the following members upon initialization:

* **double CAKE\_GRAMS = 250**
* **double CAKE\_CALORIES = 1000**
* **BigDecimal CAKE\_PRICE = 5**

A **Salmon** must have the following members upon initialization:

* **double SALMON\_GRAMS = 22**

Zip your package and upload it to Judge.

## Animals

NOTE: You need a public class **Main**.

Create a hierarchy (package) of **animals**. Your program should have three different animals – **Dog**, **Frog,** and **Cat**. Deeper in the hierarchy you should have two additional classes – **Kitten** and **Tomcat**. **Kittens are "Female" and Tomcats are "Male".** All types of animals should be able to produce some kind of sound - String produceSound**()**. For example, the dog should be able to bark. Your task is to model the hierarchy and test its functionality. Create an animal of each kind and make them all produce sound and create getters for all fields.

You will be given some lines of input. Every two lines will represent an animal. On the first line will be the type of animal and on the second – the name, the age, and the gender. When the command "**Beast!**" is given, stop the input and print all the animals in the format shown below.

### Output

* Print the information for each animal in three lines. On the first line, print: **"{animalType}"**.
* On the second line, print: **"{name} {age} {gender}"**.
* On the third line, print the sounds it produces: **"{produceSound()}"**.

### Constraints

* Each **Animal** should have a **name**, an **age,** and a **gender.**
* **All** input values should **not be blank** (e.g. name, age, and so on…).
* If you receive an input for the **gender** of a **Tomcat** or a **Kitten**, ignore it but **create** the animal.
* If the input is invalid for one of the properties, throw an exception with the message: **"Invalid input!"**.
* Each animal should have the functionality to **produceSound()**.
* Here is the type of sound each animal should produce:
  + Dog: **"Woof!"**
  + Cat: **"Meow meow"**
  + Frog: **"Ribbit"**
  + Kittens: **"Meow"**
  + Tomcat: **"MEOW"**

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Cat  Tom 12 Male  Dog  Rex 132 Male  Beast! | Cat  Tom 12 Male  Meow meow  Dog  Rex 132 Male  Woof! |
| Frog  Kermit 12 Male  Beast! | Frog  Kermit 12 Male  Ribbit |
| Frog  Froakie -2 Male  Frog  Froakie 2 Male  Beast! | Invalid input!  Frog  Froakie 2 Male  Ribbit |

### Hint

To find the name of the class you can use this.getClass().getSimpleName() in toString() method inside Animal class.