# Lab: Abstraction and Polymorphism

Tasks for exercise in class and for homework to the course  ["Programming Advanced for QA" @ SoftUni](https://softuni.bg/trainings/4257/programming-advanced-for-qa-november-2023).

Test your tasks in the Judge system: <https://judge.softuni.org/Contests/4465/Abstraction-Polymorphism-Lab>

# Shapes

Build a **hierarchy** of **interfaces** and **classes**:



You should be able to use the class like this:

|  |
| --- |
| Program.cs |
| int radius = int.Parse(Console.ReadLine()!);  IDrawable circle = new Circle(radius);  int width = int.Parse(Console.ReadLine()!);  int height = int.Parse(Console.ReadLine()!);  IDrawable rectangle = new Rectangle(width, height);  circle.Draw();  rectangle.Draw(); |

## Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3  4  5 | \*\*\*\*\*\*\*  \*\* \*\*  \*\* \*\*  \* \*  \*\* \*\*  \*\* \*\*  \*\*\*\*\*\*\*  \*\*\*\*  \* \*  \* \*  \* \*  \*\*\*\* |

## Hints

The algorithm for drawing a circle is:

A computer screen shot of a code

Description automatically generated

The algorithm for drawing a rectangle is:

A screenshot of a computer program

Description automatically generated

# Cars

Build a **hierarchy** of **interfaces** and **classes**:



Your hierarchy must be used with this code:

|  |
| --- |
| StartUp.cs |
| ICar seat = new Seat("Leon", "Grey");  ICar tesla = new Tesla("Model 3", "Red", 2);  Console.WriteLine(seat.ToString());  Console.WriteLine(tesla.ToString()); |

## Examples

|  |
| --- |
| **Output** |
| Grey Seat Leon  Engine start  Break!  Red Tesla Model 3 with 2 Batteries Engine start  Break! |

# Animals

**NOTE**: You need a **folder** named Models to hold the **classes** in.

Create an **abstract** class Animal, which holds two fields:

* name: **string**
* favouriteFood: **string**

An animal has one **virtual** **method** ExplainSelf()**: string.**You should add two new classes - **Cat** and **Dog. Override** the ExplainSelf() method by adding **concrete** **animal** **sound** on a new line. (Look at examples below)

You should be able to use the class like this:

|  |
| --- |
| Program.cs |
| Animal cat = new Cat("Peter", "Whiskas");  Animal dog = new Dog("George", "Meat");  Console.WriteLine(cat.ExplainSelf());  Console.WriteLine(dog.ExplainSelf()); |

## Examples

|  |
| --- |
| **Output** |
| I am Peter and my fovourite food is Whiskas  MEEOW  I am George and my fovourite food is Meat  BORK |

# Raiding

Your task is to create a class hierarchy like the one shown here:

A screenshot of a computer

Description automatically generated

The **BaseHero** class should be **abstract** and inherit fromIHero.

* **BaseHero – abstract – has string Name, abstract int Power, virtual string CastAbility():** {this.GetType().Name} - {this.Name}
* **Fighter – abstract – inherits BaseHero, overrides CastAbility(): {base.CastAbility()} hit for {this.Power} damage**
* **Healer – abstract – inherits BaseHero, overrides CastAbility(): {base.CastAbility()} healed for {this.Power}**

Now create concrete classes:

* + **Druid – inherits Healer, overrides power = 80, overrides CastAbility(): {base.CastAbility()}**
  + **Paladin – inherits Healer, overrides power = 100, overrides CastAbility(): {base.CastAbility()}**
  + **Rogue – inherits Fighter, overrides power = 80, overrides CastAbility(): {base.CastAbility()}**
  + **Warrior – inherits Fighter, overrides power = 100, overrides CastAbility(): {base.CastAbility()}**

## Example

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3  Mike  Paladin  Josh  Druid  Scott  Warrior  250 | Paladin - Mike healed for 100  Druid - Josh healed for 80  Warrior - Scott hit for 100 damage  Victory! |
| 2  Mike  Warrior  Tom  Rogue  200 | Warrior - Mike hit for 100 damage  Rogue - Tom hit for 80 damage  Defeat... |