# Lab: Interfaces and Abstraction

Problems for exercises and homework for the ["Java OOP Advanced" course @ SoftUni.](https://softuni.bg/courses/java-oop-advanced)

You can check your solutions here: <https://judge.softuni.bg/Contests/498/Interfaces-and-Abstraction-Lab> .

## Shapes Drawing

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

|  |
| --- |
| <<Drawable>> |
| **+draw()** |

|  |
| --- |
| **Circle** |
| **-radius: Integer** |

|  |
| --- |
| Rectangle |
| **-width: Integer**  **-height: Integer** |

You should be able to use the class like this:

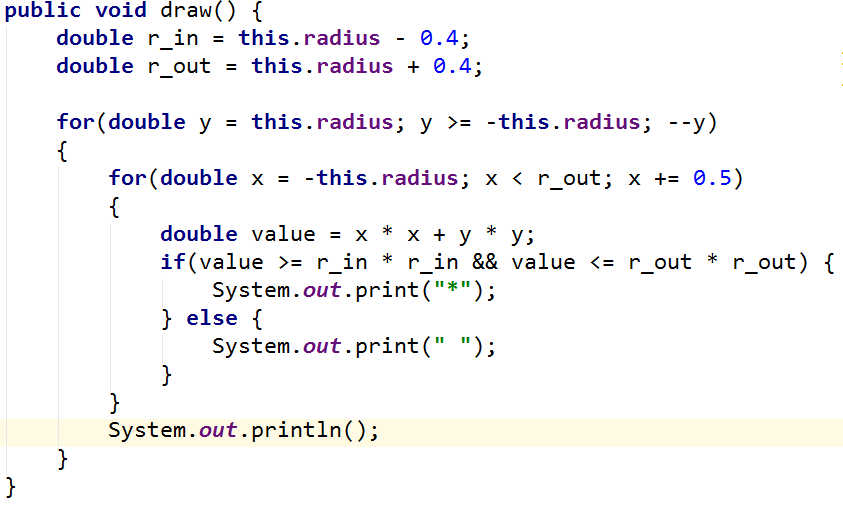
|  |
| --- |
| Main.java |
| **public static void** main(String[] args) {  Scanner scanner = **new** Scanner(System.***in***);  Queue<Integer> queue = **new** ArrayDeque<>();  **for** (**int** i = 0; i < 5; i++) {  queue.add(Integer.*parseInt*(scanner.nextLine()));  }   Drawable circle = **new** Circle(queue.poll());  Drawable rect = **new** Rectangle(queue.poll());   circle.draw();  rect.draw(); } |

### Examples

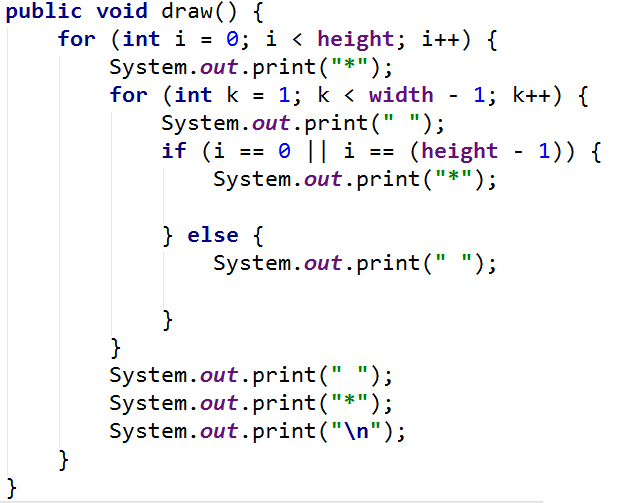
|  |  |
| --- | --- |
| **Input** | **Output** |
| 4  6  6  5  4 | \*\*\*\*\*\*\*  \*\*\* \*\*\*  \*\* \*\*  \*\* \*\*  \* \*  \*\* \*\*  \*\* \*\*  \*\*\* \*\*\*  \*\*\*\*\*\*\*  \* \* \* \* \*  \* \*  \* \*  \* \* \* \* \* |

### Solution

For **circle** drawing you can use this algorithm:



For **rectangle** drawing algorithm will be:



## Car Shop

Build hierarchy from **classes** and **interfaces** for this UML diagram

|  |
| --- |
| <<inteface>>  <<Car>> |
| **+TIRES: Integer** |
| **+getModel(): String**  **+getColor(): String**  **+getHorsePower(): Integer** |

|  |
| --- |
| **<<Serializable>>** |

|  |
| --- |
| Seat |
| **-countryProduced: String** |
| **+toString(): String** |

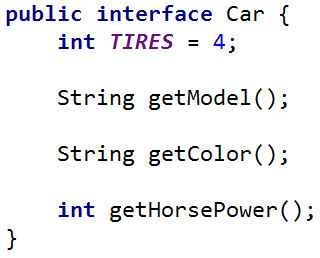
Your hierarchy have to be used with this code

|  |
| --- |
| Main.java |
| **public static void** main(String[] args) {  Car seat = **new** Seat(**"Leon"**, **"gray"**, 110, **"Spain"**);   System.***out***.println(String.*format*(  **"%s is %s color and have %s horse power"**,  seat.getModel(),  seat.getColor(),  seat.getHorsePower()));  System.***out***.println(seat.toString()); } |

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
|  | Leon is gray and have 110 horse power  This is Leon produced in Spain and have 4 tires |

### Solution



**Note:** consider using the wrapper classes in the **Seat** constructor**.**

## Car Shop Extend

|  |
| --- |
| **<<Car>>** |

Extend previous problem:

|  |
| --- |
| <<Rentable>> |
| **-minRentDay: Integer**  **-pricePerDay: Double** |
| **+getMinRentDay(): Integer**  **+getPricePerDay(): Double** |

|  |
| --- |
| <<Sellable>> |
| **-price: Double** |
| **+getPrice(): Double** |

|  |
| --- |
| Seat |
| **-countryProduced: String** |
| **+toString(): String** |

|  |
| --- |
| Audi |
| **-countryProduced: String** |
| **+toString(): String** |

Your hierarchy have to be used with this code

|  |
| --- |
| Main.java |
| **public static void** main(String[] args) {  Sellable seat = **new** Seat(**"Leon"**, **"Gray"**, 110, **"Spain"**, 11111.1);  Rentable audi = **new** Audi(**"Leon"**, **"Gray"**, 110, **"Spain"**, 3, 99.9);   *printCarInfo*(seat);  *printCarInfo*(audi); }  **private static void** printCarInfo(Car car) {  System.***out***.println(String.*format*(  **"%s is %s color and have %s horse power"**,  car.getModel(),  car.getColor(),  car.getHorsePower()));  System.***out***.println(car.toString()); } |

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
|  | Leon is gray and have 110 horse power  This is Leon produced in Spain and have 4 tires |

## Say Hello

Build hierarchy from classes and interfaces for this UML diagram

|  |
| --- |
| <<Person>> |
| **+getName(): String**  **+sayHello(): String** |

|  |
| --- |
| Chinese |
| **-name: String** |
| **-setName(): void** |

|  |
| --- |
| Bulgarian |
| **-name: String** |
| **-setName(): void** |

|  |
| --- |
| European |
| **-name: String** |
| **-setName(): void** |

Your hierarchy have to be used with this code

|  |
| --- |
| Main.java |
| **public static void** main(String[] args) {  List<Person> persons = **new** ArrayList<>();  persons.add(**new** Bulgarian(**"Pesho"**));  persons.add(**new** European(**"Pesho"**));  persons.add(**new** Chinese(**"Pesho"**));   **for** (Person person : persons) {  *print*(person);  } }  **private static void** print(Person person) {  person.sayHello(); } |

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
|  | Здравей  Hello  Djydjybydjy |

## Say Hello Extend

Build hierarchy from classes and interfaces for this UML diagram

|  |
| --- |
| Bulgarian |
| **+sayHello(): String** |

|  |
| --- |
| <<Person>> |
| **+getName(): String**  **+sayHello(): String** |

|  |
| --- |
| *BasePerson* |
| **-name: String** |
| **+getName(): String**  **-setName(): void** |

|  |
| --- |
| European |
| +sayHello(): String |

|  |
| --- |
| Chinese |
| **+sayHello(): String** |

Your hierarchy have to be used with this code

|  |
| --- |
| Main.java |
| **public static void** main(String[] args) {  List<Person> persons = **new** ArrayList<>();  persons.add(**new** Bulgarian(**"Pesho"**));  persons.add(**new** European(**"Pesho"**));  persons.add(**new** Chinese(**"Pesho"**));   **for** (Person person : persons) {  *print*(person);  } }  **private static void** print(Person person) {  person.sayHello(); } |

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
|  | Здравей  Hello  Djydjybydjy |