

Міністерство освіти і науки України  
Національний технічний університет України «Київський політехнічний інститут  
імені Ігоря Сікорського»  
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 6 з дисципліни  
«Розробка програмного забезпечення на платформі Java»

«Наслідування та поліморфізм»

Варіант 5

Виконав студент

ІП-15, Буяло Дмитро Олександрович  
(шифр, прізвище, ім'я, по батькові)

Перевірив

Ковальчук Олександр Миронович  
(прізвище, ім'я, по батькові)

Київ 2023

## Лабораторна робота 6

### Наслідування та поліморфізм

**Мета** – ознайомитися з механізмом наслідування та принципом поліморфізму. Використання механізму наслідування та принципу поліморфізму в мові Java. Здобуття навичок у використанні механізму наслідування та принципу поліморфізму.

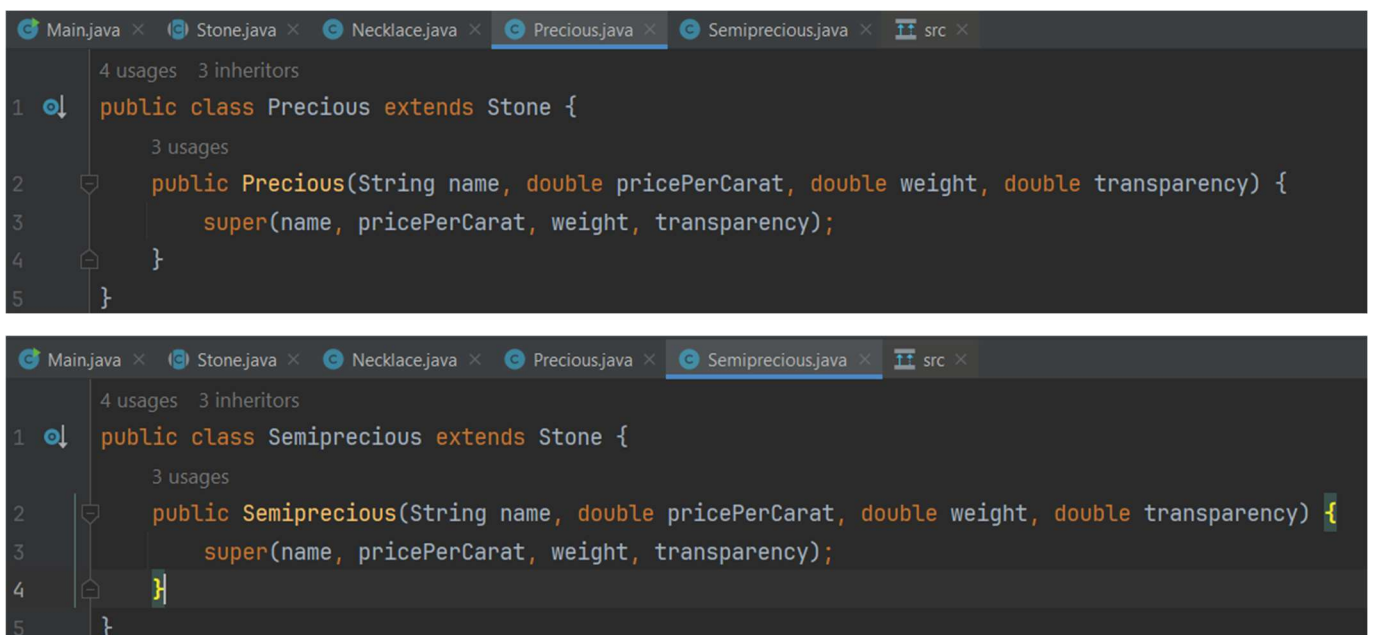
#### Індивідуальне завдання

#### Варіант 5

#### Завдання

Визначити ієрархію дорогоцінного та напівкоштовного каміння. Відібрати камені для намиста. Порахувати загальну вагу (у каратах) і вартість намиста. Провести сортування каміння намиста за цінністю. Знайти каміння в намисто, що відповідає заданому діапазону параметрів прозорості. Створити узагальнений клас та не менше 3 класів-нащадків. Необхідно обробити всі виключні ситуації, що можуть виникнути під час виконання програмного коду. Всі змінні повинні бути описані та значення їх задані у виконавчому методі.

#### Програмна реалізація



```
4 usages 3 inheritors
1 public class Precious extends Stone {
2     3 usages
3     public Precious(String name, double pricePerCarat, double weight, double transparency) {
4         super(name, pricePerCarat, weight, transparency);
5     }
6 }
```

```
4 usages 3 inheritors
1 public class Semiprecious extends Stone {
2     3 usages
3     public Semiprecious(String name, double pricePerCarat, double weight, double transparency) {
4         super(name, pricePerCarat, weight, transparency);
5     }
6 }
```

## Розробка програмного забезпечення на платформі Java

```
Main.java x Stone.java x Necklace.java x Alexandrite.java x Precious.java x Semiprecious.java x src x
1      2 usages
1      public class Alexandrite extends Precious {
2      2 usages
2      public Alexandrite(double pricePerCarat, double weight, double transparency) {
3      lightbulb super( name: "Alexandrite", pricePerCarat, weight, transparency);
4      }
5      }
```

```
Main.java x Stone.java x Necklace.java x Aquamarine.java x Precious.java x Semiprecious.java x src x
1      2 usages
1      public class Aquamarine extends Precious {
2      lightbulb 2 usages
2      public Aquamarine(double pricePerCarat, double weight, double transparency) {
3      super( name: "Aquamarine", pricePerCarat, weight, transparency);
4      }
5      }
```

```
Main.java x Stone.java x Necklace.java x Diamond.java x Precious.java x Semiprecious.java x src x
1      2 usages
1      public class Diamond extends Precious {
2      lightbulb 2 usages
2      public Diamond(double pricePerCarat, double weight, double transparency) {
3      super( name: "Diamond", pricePerCarat, weight, transparency);
4      }
5      }
```

```
Main.java x Stone.java x Necklace.java x Malachite.java x Precious.java x Semiprecious.java x src x
1      2 usages
1      public class Malachite extends Semiprecious {
2      2 usages
2      public Malachite(double pricePerCarat, double weight, double transparency) {
3      super( name: "Malachite", pricePerCarat, weight, transparency);
4      }
5      }
```

```
Main.java x Stone.java x Necklace.java x Opal.java x Precious.java x Semiprecious.java x src x
1      2 usages
1      public class Opal extends Semiprecious {
2      lightbulb 2 usages
2      public Opal(double pricePerCarat, double weight, double transparency) {
3      super( name: "Opal", pricePerCarat, weight, transparency);
4      }
5      }
```

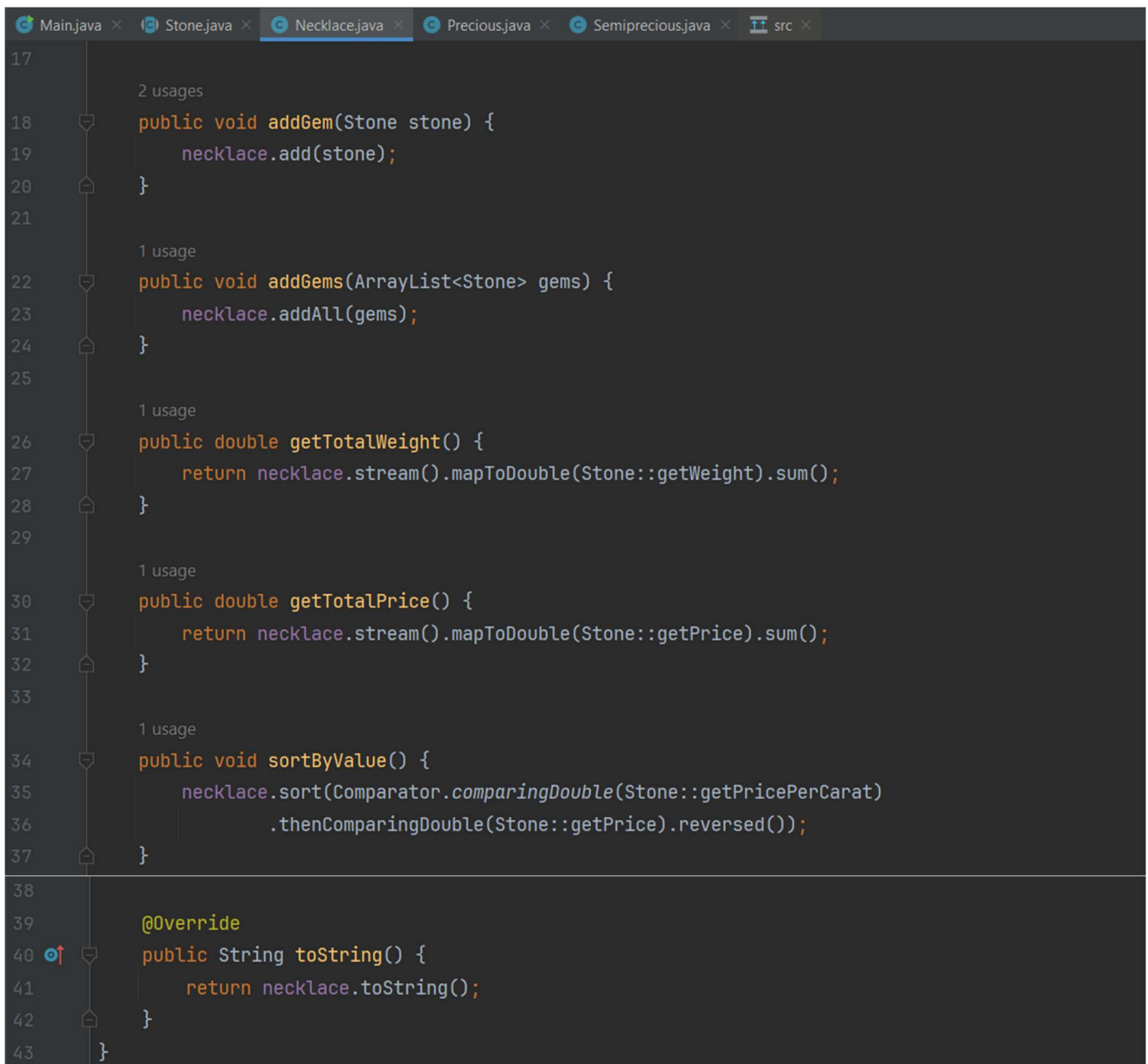
```
Main.java x Stone.java x Necklace.java x Quartz.java x Precious.java x Semiprecious.java x src x
1      1 usage
1      public class Quartz extends Semiprecious {
2      lightbulb 1 usage
2      public Quartz(double pricePerCarat, double weight, double transparency) {
3      super( name: "Quartz", pricePerCarat, weight, transparency);
4      }
5      }
```

```
Main.java x Stone.java x Necklace.java x Precious.java x Semiprecious.java x src x
23 usages 8 inheritors
1 public abstract class Stone {
    4 usages
2     private String name;
    5 usages
3     private double pricePerCarat;
    5 usages
4     private double weight;
    4 usages
5     private double transparency;
    2 usages
6     public Stone(String name, double pricePerCarat, double weight, double transparency) {
7         this.name = name;
8         this.pricePerCarat = pricePerCarat;
9         this.weight = weight;
10        this.transparency = transparency;
11    }
12
    no usages
13    public String getName() {
14        return name;
15    }
16
    no usages
17    public void setName(String name) {
18        this.name = name;
19    }
20
    3 usages
21    public double getPrice() {
22        return pricePerCarat * weight;
23    }
24
    1 usage
25    public double getPricePerCarat() {
26        return pricePerCarat;
27    }
28
    no usages
29    public void setPricePerCarat(double pricePerCarat) {
30        this.pricePerCarat = pricePerCarat;
31    }
32
    public double getWeight() {
33        return weight;
34    }
35 }
```

## Розробка програмного забезпечення на платформі Java

```
Main.java x Stone.java x Necklace.java x Precious.java x Semiprecious.java x src x
36
37     public void setWeight(double weight) {
38         this.weight = weight;
39     }
40
41     2 usages
42     public double getTransparency() {
43         return transparency;
44     }
45
46     no usages
47     public void setTransparency(double transparency) {
48         this.transparency = transparency;
49     }
50
51     @Override
52     public String toString() {
53         return name + ": price per carat = " + pricePerCarat + "€; weight = " + weight + " carats; " +
54             "total price = " + getPrice() + "€; transparency = " + transparency;
55     }
56 }
```

```
Main.java x Stone.java x Necklace.java x Precious.java x Semiprecious.java x src x
1  import java.util.ArrayList;
2  import java.util.Comparator;
3
4  5 usages
5  public class Necklace {
6      9 usages
7      private final ArrayList<Stone> necklace;
8      1 usage
9      public Necklace(ArrayList<Stone> necklace) {
10         this.necklace = necklace;
11     }
12
13     1 usage
14     public Necklace() {
15         this.necklace = new ArrayList<>();
16     }
17
18     2 usages
19     public ArrayList<Stone> getNecklace() {
20         return necklace;
21     }
22 }
```

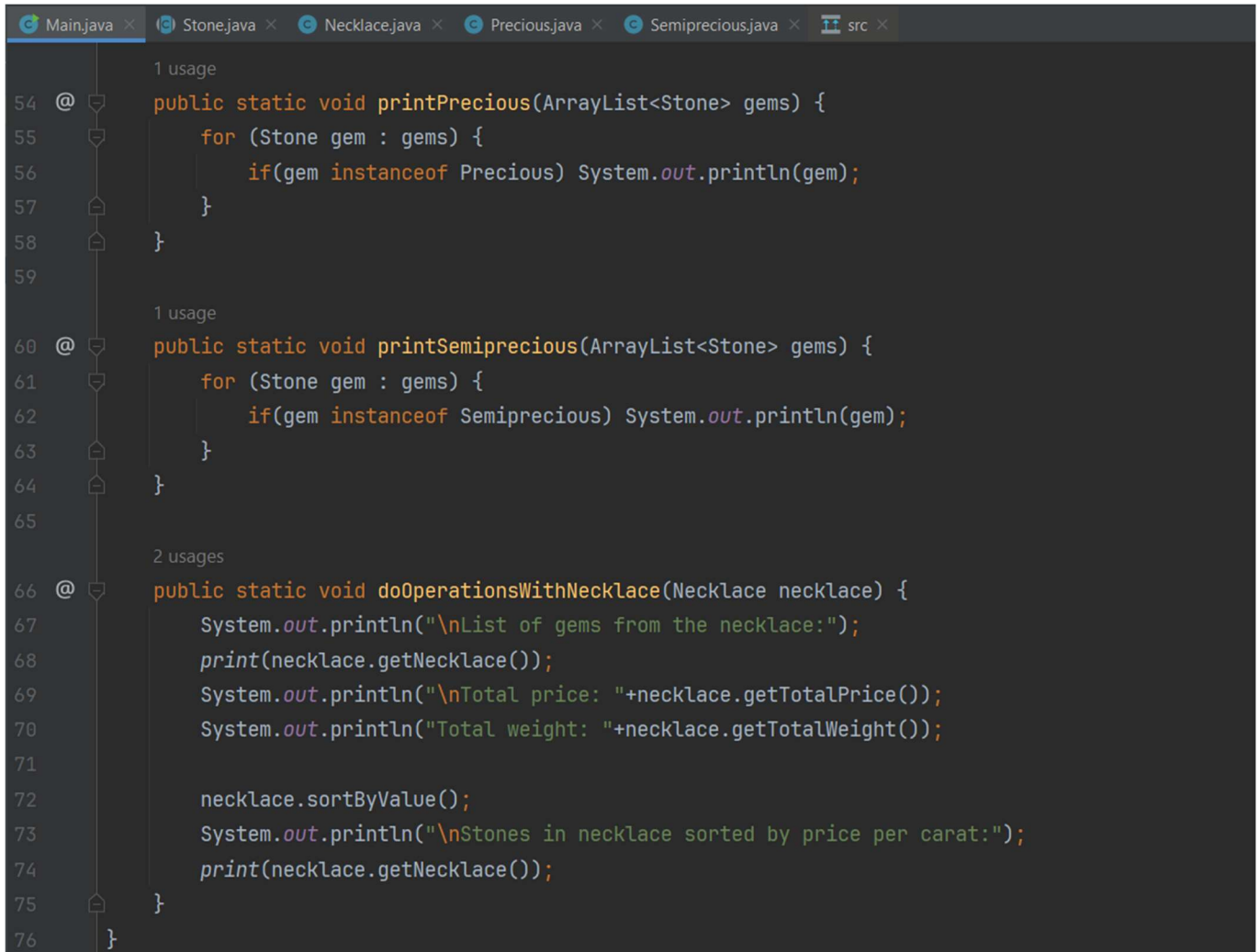


```
17  
18 2 usages  
19 public void addGem(Stone stone) {  
20     necklace.add(stone);  
21 }  
22 1 usage  
23 public void addGems(ArrayList<Stone> gems) {  
24     necklace.addAll(gems);  
25 }  
26 1 usage  
27 public double getTotalWeight() {  
28     return necklace.stream().mapToDouble(Stone::getWeight).sum();  
29 }  
30 1 usage  
31 public double getTotalPrice() {  
32     return necklace.stream().mapToDouble(Stone::getPrice).sum();  
33 }  
34 1 usage  
35 public void sortByValue() {  
36     necklace.sort(Comparator.comparingDouble(Stone::getPricePerCarat)  
37         .thenComparingDouble(Stone::getPrice).reversed());  
38 }  
39 @Override  
40 public String toString() {  
41     return necklace.toString();  
42 }  
43 }
```



## Розробка програмного забезпечення на платформі Java

```
1  import java.util.ArrayList;
2
3  public class Main {
4      public static void main(String[] args) {
5          ArrayList<Stone> gems = new ArrayList<>();
6          gems.add(new Alexandrite( pricePerCarat: 20000, weight: 1.5, transparency: 0.2));
7          gems.add(new Aquamarine( pricePerCarat: 1100, weight: 3, transparency: 0.2));
8          gems.add(new Diamond( pricePerCarat: 15800, weight: 1.9, transparency: 0.1));
9          gems.add(new Diamond( pricePerCarat: 4040, weight: 3, transparency: 0.18));
10         gems.add(new Malachite( pricePerCarat: 3, weight: 7, transparency: 0.9));
11         gems.add(new Opal( pricePerCarat: 6500, weight: 2, transparency: 0.4));
12         gems.add(new Opal( pricePerCarat: 7, weight: 5, transparency: 0.85));
13         gems.add(new Quartz( pricePerCarat: 3, weight: 2.5, transparency: 0.9));
14         gems.add(new Malachite( pricePerCarat: 3, weight: 7, transparency: 0.9));
15
16         System.out.println("Your list of gems:");
17         print(gems);
18
19         System.out.println("\nPrecious stones from the list:");
20         printPrecious(gems);
21
22         System.out.println("\nSemiprecious stones from the list:");
23         printSemiprecious(gems);
24
25         Necklace necklace = new Necklace();
26         necklace.addGem(new Aquamarine( pricePerCarat: 1100, weight: 1.7, transparency: 0.2));
27         ArrayList<Stone> gemsForNecklace = findToNecklace(gems, minTransparency: 0.05, maxTransparency: 0.4);
28         necklace.addGems(gemsForNecklace);
29         necklace.addGem(new Alexandrite( pricePerCarat: 1100, weight: 2.5, transparency: 0.2));
30
31         doOperationsWithNecklace(necklace);
32
33         System.out.println("\n!Necklace from array!");
34
35         Necklace necklaceFromArray = new Necklace(findToNecklace(gems, minTransparency: 0.35, maxTransparency: 0.99));
36         doOperationsWithNecklace(necklaceFromArray);
37     }
38
39     @ 2 usages
40     public static ArrayList<Stone> findToNecklace(ArrayList<Stone> gems,
41                                                 double minTransparency, double maxTransparency) {
42         ArrayList<Stone> fits = new ArrayList<>();
43         for (Stone gem : gems) {
44             if(gem.getTransparency()>=minTransparency && gem.getTransparency()<=maxTransparency){
45                 fits.add(gem);
46             }
47         }
48         return fits;
49
50     @ 3 usages
51     public static void print(ArrayList<Stone> gems) {
52         for (Stone gem : gems) System.out.println(gem);
53     }
```



The screenshot shows an IDE with several tabs: Main.java, Stone.java, Necklace.java, Precious.java, Semiprecious.java, and src. The Main.java file is active, showing three methods. The first method, printPrecious, iterates over an ArrayList of Stone objects and prints those that are instances of Precious. The second method, printSemiprecious, iterates over the same ArrayList and prints those that are instances of Semiprecious. The third method, doOperationsWithNecklace, prints the list of gems from a necklace, the total price, and the total weight, then sorts the necklace by value and prints the sorted list. The code is annotated with Javadoc comments and includes line numbers from 54 to 76.

```
1 usage
54 @
55 public static void printPrecious(ArrayList<Stone> gems) {
56     for (Stone gem : gems) {
57         if(gem instanceof Precious) System.out.println(gem);
58     }
59
1 usage
60 @
61 public static void printSemiprecious(ArrayList<Stone> gems) {
62     for (Stone gem : gems) {
63         if(gem instanceof Semiprecious) System.out.println(gem);
64     }
65
2 usages
66 @
67 public static void doOperationsWithNecklace(Necklace necklace) {
68     System.out.println("\nList of gems from the necklace:");
69     print(necklace.getNecklace());
70     System.out.println("\nTotal price: "+necklace.getTotalPrice());
71     System.out.println("Total weight: "+necklace.getTotalWeight());
72
73     necklace.sortByValue();
74     System.out.println("\nStones in necklace sorted by price per carat:");
75     print(necklace.getNecklace());
76 }
```

### Приклад виконання коду

Your list of gems:

Alexandrite: price per carat = 20000.0€; weight = 1.5 carats; total price = 30000.0€; transparency = 0.2  
Aquamarine: price per carat = 1100.0€; weight = 3.0 carats; total price = 3300.0€; transparency = 0.2  
Diamond: price per carat = 15800.0€; weight = 1.9 carats; total price = 30020.0€; transparency = 0.1  
Diamond: price per carat = 4040.0€; weight = 3.0 carats; total price = 12120.0€; transparency = 0.18  
Malachite: price per carat = 3.0€; weight = 7.0 carats; total price = 21.0€; transparency = 0.9  
Opal: price per carat = 6500.0€; weight = 2.0 carats; total price = 13000.0€; transparency = 0.4  
Opal: price per carat = 7.0€; weight = 5.0 carats; total price = 35.0€; transparency = 0.85  
Quartz: price per carat = 3.0€; weight = 2.5 carats; total price = 7.5€; transparency = 0.9  
Malachite: price per carat = 3.0€; weight = 7.0 carats; total price = 21.0€; transparency = 0.9

Precious stones from the list:

Alexandrite: price per carat = 20000.0€; weight = 1.5 carats; total price = 30000.0€; transparency = 0.2  
Aquamarine: price per carat = 1100.0€; weight = 3.0 carats; total price = 3300.0€; transparency = 0.2  
Diamond: price per carat = 15800.0€; weight = 1.9 carats; total price = 30020.0€; transparency = 0.1  
Diamond: price per carat = 4040.0€; weight = 3.0 carats; total price = 12120.0€; transparency = 0.18



Semiprecious stones from the list:

Malachite: price per carat = 3.0€; weight = 7.0 carats; total price = 21.0€; transparency = 0.9  
Opal: price per carat = 6500.0€; weight = 2.0 carats; total price = 13000.0€; transparency = 0.4  
Opal: price per carat = 7.0€; weight = 5.0 carats; total price = 35.0€; transparency = 0.85  
Quartz: price per carat = 3.0€; weight = 2.5 carats; total price = 7.5€; transparency = 0.9  
Malachite: price per carat = 3.0€; weight = 7.0 carats; total price = 21.0€; transparency = 0.9

List of gems from the necklace:

Aquamarine: price per carat = 1100.0€; weight = 1.7 carats; total price = 1870.0€; transparency = 0.2  
Alexandrite: price per carat = 20000.0€; weight = 1.5 carats; total price = 30000.0€; transparency = 0.2  
Aquamarine: price per carat = 1100.0€; weight = 3.0 carats; total price = 3300.0€; transparency = 0.2  
Diamond: price per carat = 15800.0€; weight = 1.9 carats; total price = 30020.0€; transparency = 0.1  
Diamond: price per carat = 4040.0€; weight = 3.0 carats; total price = 12120.0€; transparency = 0.18  
Opal: price per carat = 6500.0€; weight = 2.0 carats; total price = 13000.0€; transparency = 0.4  
Alexandrite: price per carat = 1100.0€; weight = 2.5 carats; total price = 2750.0€; transparency = 0.2

Total price: 93060.0

Total weight: 15.6

Stones in necklace sorted by price per carat:

Alexandrite: price per carat = 20000.0€; weight = 1.5 carats; total price = 30000.0€; transparency = 0.2  
Diamond: price per carat = 15800.0€; weight = 1.9 carats; total price = 30020.0€; transparency = 0.1  
Opal: price per carat = 6500.0€; weight = 2.0 carats; total price = 13000.0€; transparency = 0.4  
Diamond: price per carat = 4040.0€; weight = 3.0 carats; total price = 12120.0€; transparency = 0.18  
Aquamarine: price per carat = 1100.0€; weight = 3.0 carats; total price = 3300.0€; transparency = 0.2  
Alexandrite: price per carat = 1100.0€; weight = 2.5 carats; total price = 2750.0€; transparency = 0.2  
Aquamarine: price per carat = 1100.0€; weight = 1.7 carats; total price = 1870.0€; transparency = 0.2

!Necklace from array!

List of gems from the necklace:

Malachite: price per carat = 3.0€; weight = 7.0 carats; total price = 21.0€; transparency = 0.9  
Opal: price per carat = 6500.0€; weight = 2.0 carats; total price = 13000.0€; transparency = 0.4  
Opal: price per carat = 7.0€; weight = 5.0 carats; total price = 35.0€; transparency = 0.85  
Quartz: price per carat = 3.0€; weight = 2.5 carats; total price = 7.5€; transparency = 0.9  
Malachite: price per carat = 3.0€; weight = 7.0 carats; total price = 21.0€; transparency = 0.9

Total price: 13084.5

Total weight: 23.5

Stones in necklace sorted by price per carat:

Opal: price per carat = 6500.0€; weight = 2.0 carats; total price = 13000.0€; transparency = 0.4  
Opal: price per carat = 7.0€; weight = 5.0 carats; total price = 35.0€; transparency = 0.85  
Malachite: price per carat = 3.0€; weight = 7.0 carats; total price = 21.0€; transparency = 0.9  
Malachite: price per carat = 3.0€; weight = 7.0 carats; total price = 21.0€; transparency = 0.9  
Quartz: price per carat = 3.0€; weight = 2.5 carats; total price = 7.5€; transparency = 0.9

Process finished with exit code 0

### **Висновок**

Під час виконання шостої лабораторної роботи, розглянули на практиці роботу з механізмом наслідування та поліморфізму, використавши їх для розробки застосунку. В результаті була створена програма, яка відповідно до варіанту, визначивши ієрархію дорогоцінного та напівкоштовного каміння, відбрає камені для намиста, рахує загальну вагу (у каратах) і вартість намиста, проводить сортування каміння намиста за цінністю, знаходить каміння в намисто, що відповідає заданому діапазону параметрів прозорості.