Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра ЕОМ

Звіт

З лабораторної роботи №7

Варіант – 21

З дисципліни: «Кросплатформенні засоби програмування»

На тему: «Параметризоване програмування»

Виконав: ст. гр. КІ-35

Сухан Д. В.

Перевірив: доцент кафедри ЕОМ

Іванов Ю.С.

Львів 2022

Мета: Оволодіти навиками параметризованого програмування мовою Java.

Завдання (варіант № 21)

21. Бак для сміття

1. Створити параметризований клас, що реалізує предметну область задану варіантом. Клас має містити мінімум 4 методи опрацювання даних включаючи розміщення та виймання елементів. Парні варіанти реалізують пошук мінімального елементу, непарні – максимального. Написати на мові Java та налагодити програму-драйвер для розробленого класу, яка мстить мінімум 2 різні класи екземпляри яких розмішуються у екземплярі розробленого класу-контейнеру. Програма має розміщуватися в пакеті Група.Прізвище.Lab6 та володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету.

2. Автоматично згенерувати документацію до розробленого пакету.

3. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації.

4. Дати відповідь на контрольні запитання.

Хід роботи:

Текст програми

Item.java

```
package KI35.Sukhan.Lab7;
/**
* Інтерфейс Item реалізує методи , які використовують
похідні класи
*
* @author Sukhan Denys
* @version 1.0
* @since version 1.0
*
*/
public interface Item extends Comparable<Item> {
    public int getSize();
    public void print();
```

```
}
```

GarbageCan.java

```
package KI35.Sukhan.Lab7;
import java.util.*;
/**
* Клас GarbageCan реалізує Бак для сміття
*
* @author Sukhan Denys
* @version 1.0
* @since version 1.0
*
*/
public class GarbageCan<T extends Item> {
    private ArrayList<T> arr;
    /**
    *Constructor
    */
    public GarbageCan() {
        arr = new ArrayList<T>();
    }

    /**
     * Method шукає максимальний розмір
     * @return T
     */
    public T findMax() {
        if (!arr.isEmpty()) {
            T max = arr.get(0);
                for (int i=1; i< arr.size(); i++) {
                    if ( arr.get(i).compareTo(max) > 0 )
max = arr.get(i);
                }
            return max;
        }
        return null;
```

```
    }


    /**
     * Method додає річ до Баку для сміття
     * @param data
     */
    public void addItem(T data) {
        arr.add(data);
        System.out.print("This Item is added to the
Garbage can: ");
        data.print();
    }


    /**
     * Method видаляє річ від Баку для сміття
     * @param i
     */
    public void deleteItem(int i) {
        System.out.print("This Item is deleted from the
Garbage can: ");
        arr.get(i).print();
        arr.remove(i);
    }

    /**
     * Method показує усі речі з  Баку для сміття
     */
    public void showUnits() {
        if (!arr.isEmpty()) {
            System.out.print("Garbage can Items :\n");
            for (int i=0; i< arr.size(); i++) {
                arr.get(i).print();
            }
        }
```

```
        else  System.out.print("Garbage can hasn't
Items.\n");
    }
}
```

CanForPlastic.java

```
package KI35.Sukhan.Lab7;
/**
* Клас CanForPlastic реалізує ріл із пластику
*
* @author Sukhan Denys
* @version 1.0
* @since version 1.0
*
*/
public class CanForPlastic implements Item {
    private String PName;
    private int size;


    /**
    *Constructor
    *@param PName
    *@param size
    */
    public CanForPlastic(String PName, int size) {
        this.PName = PName;
        this.size = size;
    }


    /**
     * Method повертає назву речі зі пластику
     * @return String
     */
    public String getPName() {
        return PName;
    }
```

```
    /**
     * Method встановлює назву речі зі пластику
     * @param buildingName
     */
    public void SetPName(String PName) {
        this.PName= PName;
    }


    /**
     * Method встановлює розмір речі зі пластику
     * @param area
     */
    public void setSize(int size) {
        this.size = size;
    }


    /**
     * Method повертає розмір речі зі пластику
     * @return int
     */
    public int getSize() {
        return size;
    }


    /**
     * Method порівнює розімр речі зі пластику з розіміром
речі іншого класу
     * @param p
     * @return int
     */
    public int compareTo(Item p) {
        Integer s = size;
```

```
        return s.compareTo(p.getSize());
    }

    /**
     * Method виводить інформацію про річ з пластику
     */
    public void print() {
        System.out.print("Plastic thing name:  - "+ PName
+ ". Its size : "+ size + " \n");
    }
}
```

CanForGlass.html

```
package KI35.Sukhan.Lab7;
/**
* Клас CanForGlass реалізує річ зі скла
*
* @author Sukhan Denys
* @version 1.0
* @since version 1.0
*
*/
public class CanForGlass implements Item {
    private String GName;
    private int size;

    /**
    *Constructor
    *@param GName
    *@param size
    */
    public CanForGlass(String GName, int size) {
        this.GName = GName;
        this.size = size;
    }

    /**
```

```
     * Method повертає назву речі зі скла
     * @return String
     */
    public String getGName() {
        return GName;
    }


    /**
     * Method встановлює назву речі зі скла
     * @param GName
     */
    public void setGName(String GName) {
        this.GName = GName;
    }


    /**
     * Method встановлює розмір для речі зі скла
     * @param size
     */
    public void setSize(int size) {
        this.size = size;
    }


    /**
     * Method повертає розмір  речі зі скла
     * @return int
     */
    public int getSize() {
        return size;
    }


    /**
```

```
     * Method порівнює розімр речі зі скла з розміром
речі  іншого класу
     * @param p
     * @return int
     */
    public int compareTo(Item p) {
        Integer s = size;
        return s.compareTo(p.getSize());
    }


    /**
     * Method виводить інформацію річ зі скла
     */
    public void print() {
        System.out.print("Glass thing name:  - "+ GName +
". Its size : "+ size + " \n");
    }

}
```

App.html

```
package KI35.Sukhan.Lab7;
/**
* Клас App реалізує програму-драйвер
*
* @author Sukhan Denys
* @version 1.0
* @since version 1.0
*
*/
public class App {
    /**
     * @param args
     */
    public static void main(String[] args) {
```

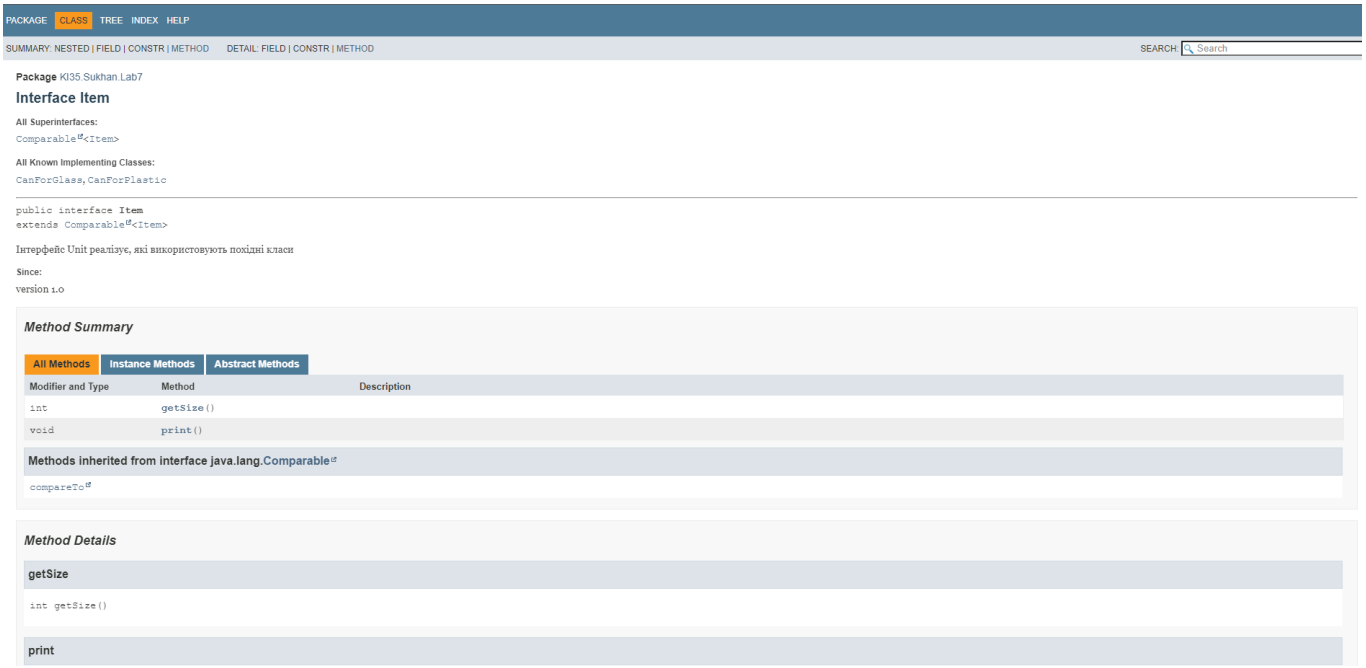
```
        GarbageCan <? super Item> garbageCan = new
GarbageCan <Item>();

        garbageCan.addItem(new CanForGlass("Plate", 20));
        garbageCan.addItem(new CanForPlastic("Bottle" ,
35));
        garbageCan.addItem(new CanForPlastic("Chair" ,
68));
        garbageCan.addItem(new CanForGlass("Window", 75));

        Item res = garbageCan.findMax();
        System.out.print("The max size in Garbage can :
\n");
        res.print();
    }
}
```

Фрагмент згенерованої документації

Item.html



PACKAGE CLASS TREE INDEX HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD SEARCH: Search

Package KI35.Sukhan.Lab7

Interface Item

All Superinterfaces:
Comparable<Item>

All Known Implementing Classes:
CanForGlass, CanForPlastic

public interface Item
extends Comparable<Item>

Інтерфейс Unit реалізує, які використовують похідні класи

Since:
version 1.0

Method Summary

All Methods | Instance Methods | Abstract Methods

Modifier and Type	Method	Description
int	getSize()	
void	print()	

Methods inherited from interface java.lang.Comparable
compareTo

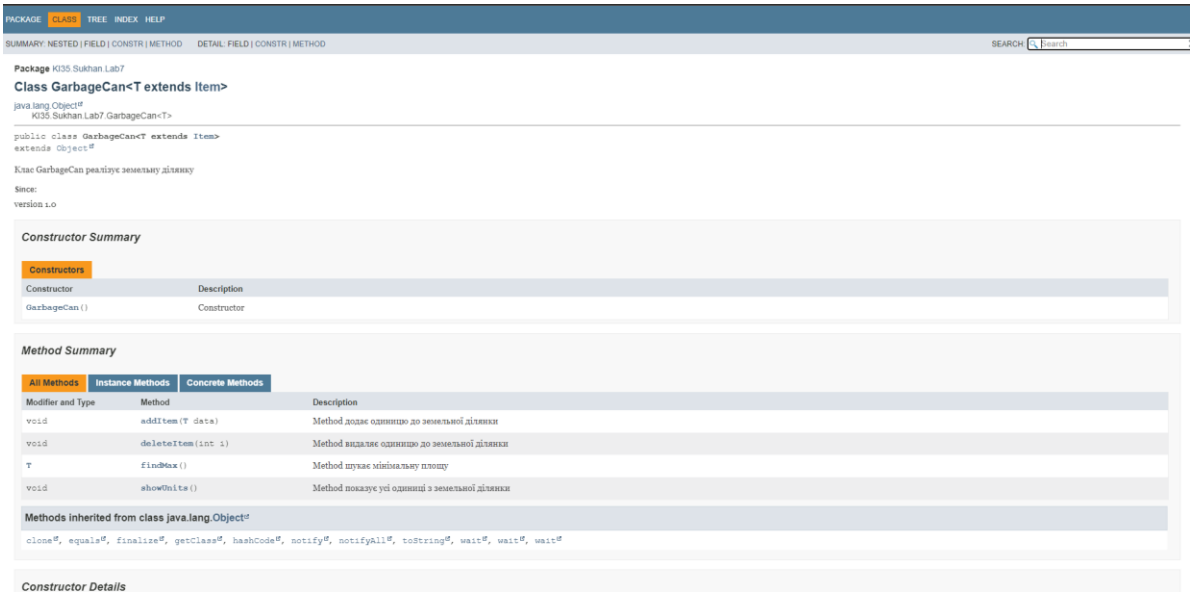
Method Details

getSize

int getSize()

print

GarbageCan.html



PACKAGE CLASS TREE INDEX HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD SEARCH: Search

Package KI35.Sukhan.Lab7

Class GarbageCan<T extends Item>

java.lang.Object
KI35.Sukhan.Lab7.GarbageCan<T>

```
public class GarbageCan<T extends Item>
extends Object
```

Клас GarbageCan реалізує земельну ділянку

Since:
version 1.0

Constructor Summary

Constructors

Constructor	Description
GarbageCan()	Constructor

Method Summary

All Methods | Instance Methods | Concrete Methods

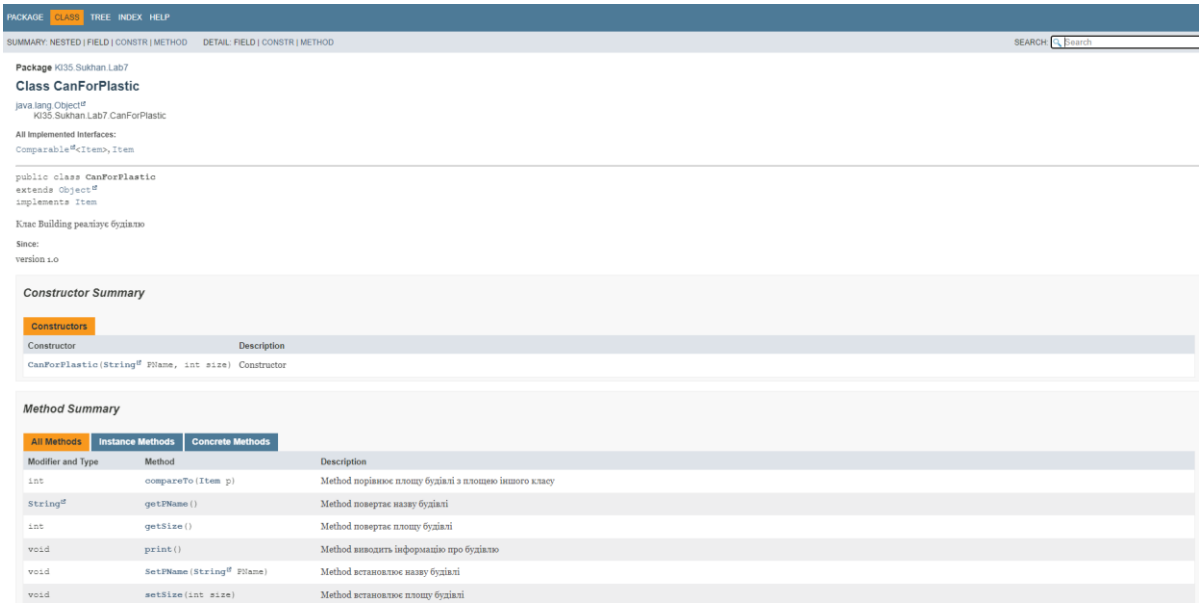
Modifier and Type	Method	Description
void	addItem(T data)	Method додає одиницю до земельної ділянки
void	deleteItem(int i)	Method видаляє одиницю до земельної ділянки
T	findMax()	Method шукає мінімальну площу
void	showUnits()	Method показує усі одиниці з земельної ділянки

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Details

CanForPlastic.html



PACKAGE CLASS TREE INDEX HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD SEARCH: Search

Package KI35.Sukhan.Lab7

Class CanForPlastic

java.lang.Object
KI35.Sukhan.Lab7.CanForPlastic

All Implemented Interfaces:
Comparable<Item>, Item

```
public class CanForPlastic
extends Object
implements Item
```

Клас Building реалізує будівлю

Since:
version 1.0

Constructor Summary

Constructors

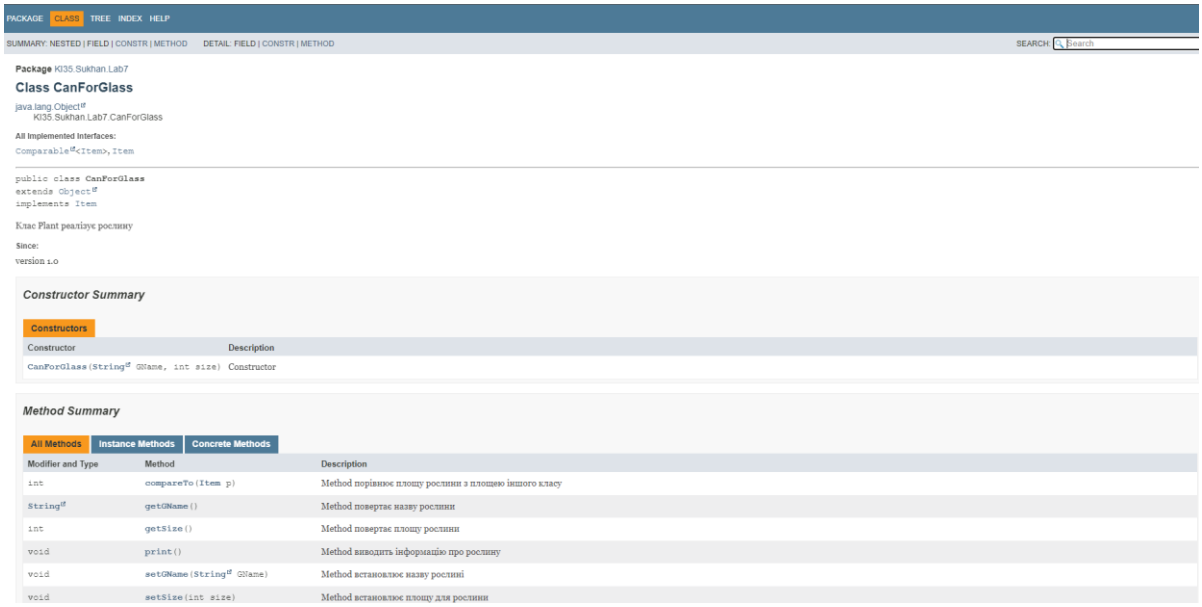
Constructor	Description
CanForPlastic(String PName, int size)	Constructor

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
int	compareTo(Item p)	Method порівнює площу будівлі з площею іншого класу
String	getPName()	Method повертає назву будівлі
int	getSize()	Method повертає площу будівлі
void	print()	Method виводить інформацію про будівлю
void	SetPName(String PName)	Method встановлює назву будівлі
void	setSize(int size)	Method встановлює площу будівлі

CanForGlass.html



PACKAGE CLASS TREE INDEX HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD SEARCH: Search

Package KI35.Sukhan.Lab7

Class CanForGlass

java.lang.Object
KI35.Sukhan.Lab7.CanForGlass

All Implemented Interfaces:
Comparable<Item>, Item

```
public class CanForGlass
extends Object
implements Item
```

Клас Plant реалізує рослину

Since:
version 1.0

Constructor Summary

Constructors

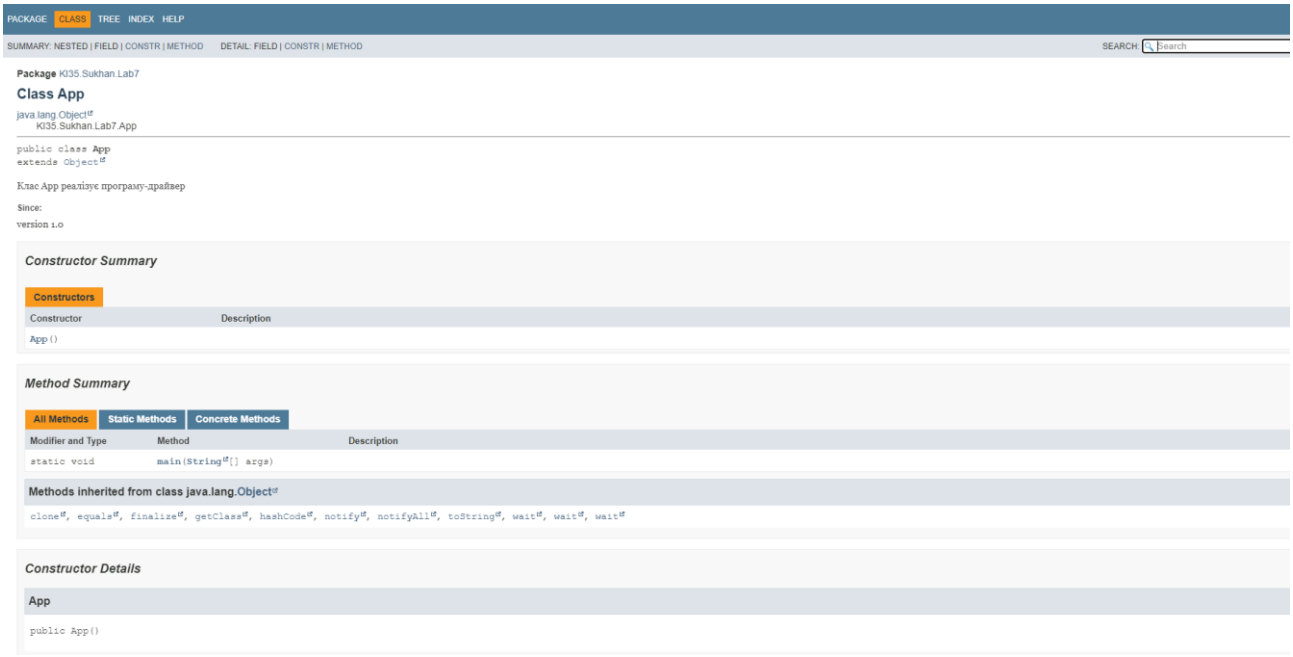
Constructor	Description
CanForGlass(String GName, int size)	Constructor

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
int	compareTo(Item p)	Method порівнює площу рослини з площею іншого класу
String	getGName()	Method повертає назву рослини
int	getSize()	Method повертає площу рослини
void	print()	Method виводить інформацію про рослину
void	setGName(String GName)	Method встановлює назву рослині
void	setSize(int size)	Method встановлює площу для рослини

App.html


PACKAGE CLASS TREE INDEX HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD SEARCH: Search

Package Kl35.Sukhan.Lab7

Class App

java.lang.Object
Kl35.Sukhan.Lab7.App

public class App
extends Object

Клас App реалізує програму-драйвер

Since:
version 1.0

Constructor Summary

Constructors

Constructor	Description
App()	

Method Summary

All Methods | Static Methods | Concrete Methods

Modifier and Type	Method	Description
static void	main(String[] args)	

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Details

App

public App()

Відповідь на контрольні запитання:

1. Дайте визначення терміну «параметризоване програмування».

Відповідь: Параметризоване програмування є аналогом шаблонів у C++. Воно полягає у написанні коду, що можна багаторазово застосовувати з об’єктами різних класів.

2. Розкрийте синтаксис визначення простого параметризованого класу.

Відповідь: Параметризований клас – це клас з однією або більше змінними типу.

[public] class НазваКласу<параметризованийТип {,параметризованийТип} > {…}

3. Розкрийте синтаксис створення об’єкту параметризованого класу.

Відповідь: НазваКласу < перелікТипів > = new НазваКласу < перелікТипів > (параметри);

4. Розкрийте синтаксис визначення параметризованого методу.

Відповідь: Параметризовані методи визначаються в середині як звичайних класів так і параметризованих.

Модифікатори< параметризованийТип{,параметризованийТип} > типПовернення назваМетоду(параметри);

5. Розкрийте синтаксис виклику параметризованого методу.

Відповідь: (НазваКласу|НазваОб’єкту).[<перелікТипів>] НазваМетоду(параметри);

6. Яку роль відіграє встановлення обмежень для змінних типів?

Відповідь: Бувають ситуації, коли клас або метод потребують накладення обмежень на змінні типів. Наприклад, може бути ситуація, коли метод у процесі роботи викликає з-під об’єкта параметризованого типу метод, що визначається у деякому інтерфейсі.

7. Як встановити обмеження для змінних типів?

Відповідь: використати ключове слово extends і вказати один суперклас, або довільну кількість інтерфейсів (через знак &), від яких має походити реальний тип, що підставляється замість параметризованого типу.

8. Розкрийте правила спадкування параметризованих типів.

Відповідь:

- Всі класи, що утворені з одного і того ж параметризованого класу з використанням різних значень змінних типів є незалежними навіть якщо між цими типами є залежність спадкування.

- Завжди можна перетворити параметризований клас у «сирий» клас.

- Параметризовані класи можуть розширювати або реалізовувати інші параметризовані класи.

9. Яке призначення підстановочних типів?

Відповідь: Підстановочні типи були введені у мову Java для збільшення гнучкості жорсткої існуючої системи параметризованих типів. На відміну від неї підстановочні типи дозволяють враховувати залежності між типами, що виступають параметрами для параметризованих типів.

10. Застосування підстановочних типів.

Відповідь:

- обмеження підтипу;

- обмеження супертипу;

- необмежені підстановки.

Висновок: Я оволодів навиками параметризованого програмування мовою Java.