

---

# Niezawodność Systemów Informatycznych

## Filip Rzepiela 4Is(s) grupa P2

---

### ZADANIA:

Zaimplementuj klasy: stos, kolejka na bazie następującego interfejsu polimorficznego (na ocenę dst):

```
0 references
abstract class Container
{
    protected int pointer = -1;
    protected int[] buffer = new int[10];

    0 references
    public abstract int pop();

    0 references
    public abstract void push(int value);
}
```

Do powyższej implementacji dołącz możliwości sprawdzania rozmiaru kolekcji, czyszczenia jej itp (metody Clear(), GetCount(), IsEmpty(), IsFull()). Staraj się to wykonać w optymalny sposób, korzystając z dziedziczenia i polimorfizmu. (na ocenę db).

### REALIZACJA ZADAŃ:

#### Zadanie 1

Zaimplementuj klasy: stos, kolejka na bazie następującego interfejsu polimorficznego.

```
abstract class Container
{
    protected int pointer = -1;
    protected int[] buffer = new int[10];
    public abstract int pop();
    public abstract void push(int value);
    public abstract void show();
}
```

```
class Stos : Container
{
    public override void push(int value)
    {
        pointer++;
        if (pointer < 10)
        {
            buffer[pointer] = value;
        }
        else
        {
            pointer--;
        }
    }

    public override int pop()
    {
        if (pointer < 0)
        {
            pointer = 0;
        }
        int tmp1 = buffer[pointer];
        pointer--;
        return tmp1;
    }

    public override void show()
    {
        Console.Write("Stos: ");
        for (int i = 0; i <= pointer; i++)
        {
            Console.Write(buffer[i]);
            Console.Write(" ");
        }
        Console.WriteLine(" ");
    }
}
```

**Metoda Pop** – usuwa ostatni element ze stosu

**Metoda Push** – wypycha elementy do stosu

## Wywołanie stosu

```
Stos stosik1 = new Stos();
stosik1.push(2);
stosik1.push(4);
stosik1.show();
stosik1.push(6);
stosik1.show();
stosik1.pop();
stosik1.show();
```

## Klasa kolejka

```
class Kolejka : Container
{
    public Kolejka()
    {

    }

    public override int pop()
    {
        int tmp = buffer[0];
        for (int i = 1; i <= pointer; i++)
        {
            buffer[i - 1] = buffer[i];
        }
        pointer--;
        if (pointer < -1)
        {
            pointer = -1;
        }
        return tmp;
    }

    public override void push(int value)
    {
        pointer++;
        if (pointer < 10)
        {
            buffer[pointer] = value;
        }
        else
        {
            pointer--;
        }
    }

    public override void show()
    {
```

```

        Console.Write("Kolejka: ");
        for (int i = 0; i <= pointer; i++)
        {
            Console.Write(buffer[i]);
            Console.Write(" ");
        }
        Console.WriteLine(" ");
    }
}

```

**Metoda Pop** – usuwa pierwszy element z kolejki

**Metoda Push** – wypycha elementy do kolejki

Wywołanie kolejki

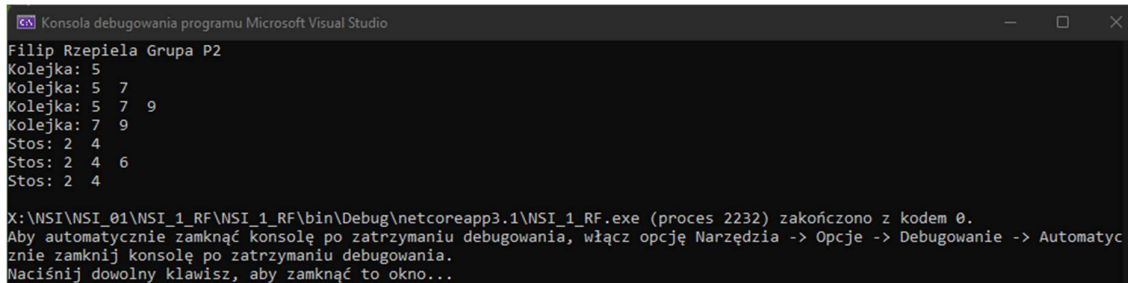
```

Kolejka kolejka1 = new Kolejka();

kolejka1.push(5);
kolejka1.show();
kolejka1.push(7);
kolejka1.show();
kolejka1.push(9);
kolejka1.show();
kolejka1.pop();
kolejka1.show();

```

Działanie kodu:



```

Konsola debugowania programu Microsoft Visual Studio
Filip Rzepiela Grupa P2
Kolejka: 5
Kolejka: 5 7
Kolejka: 5 7 9
Kolejka: 7 9
Stos: 2 4
Stos: 2 4 6
Stos: 2 4
X:\NSI\NSI_01\NSI_1_RF\NSI_1_RF\bin\Debug\netcoreapp3.1\NSI_1_RF.exe (proces 2232) zakończono z kodem 0.
Aby automatycznie zamknąć konsolę po zatrzymaniu debugowania, włącz opcję Narzędzia -> Opcje -> Debugowanie -> Automatycznie zamknij konsolę po zatrzymaniu debugowania.
Naciśnij dowolny klawisz, aby zamknąć to okno...

```

## Zadanie 2

Do powyższej implementacji dołącz możliwości sprawdzania rozmiaru kolekcji, czyszczenia jej itp (metody Clear(), GetCount(), IsEmpty(), IsFull()). Staraj się to wykonać w optymalny sposób, korzystając z dziedziczenia i polimorfizmu.

**Metoda Clear()** – wyzerowanie, usunięcie wszystkich elementów

```
public void clear()
{
    buffer = new int[10];
    pointer = -1;
}
```

**Metoda GetCount()** – zlicza ilość elementów

```
public int getcount()
{
    return pointer + 1;
}
```

**Metoda IsEmpty()** – sprawdza czy stos lub kolejka są puste

```
public Boolean isempty()
{
    if (pointer == -1)
        return true;
    else
        return false;
}
```

**Metoda IsFull()** – sprawdza czy stos lub kolejka są pełne

```
public Boolean isfull()
{
    if (pointer + 1 == 10)
        return true;
    else
        return false;
}
```

Wywołanie metod w kolejce

```
Kolejka kolejka1 = new Kolejka();
    kolejka1.push(1);
    kolejka1.push(5);
    kolejka1.show();
    kolejka1.push(7);
    kolejka1.show();
    kolejka1.push(9);
    kolejka1.show();
```

```

kolejka1.pop();
kolejka1.show();
Console.WriteLine(kolejka1.getcount());
kolejka1.push(4);
kolejka1.push(5);
kolejka1.show();
Console.WriteLine(kolejka1.getcount());
kolejka1.clear();
kolejka1.show();
Console.WriteLine(kolejka1.getcount());
Console.WriteLine(kolejka1.isempty());
kolejka1.push(2);
Console.WriteLine(kolejka1.isempty());
kolejka1.push(2);
kolejka1.push(2);
kolejka1.push(2);
kolejka1.push(2);
kolejka1.push(2);
kolejka1.push(2);
kolejka1.push(2);
kolejka1.push(2);
kolejka1.push(2);
kolejka1.show();
Console.WriteLine(kolejka1.getcount());
Console.WriteLine(kolejka1.isfull());

```

Wywołanie metod w stosie

```

Stos stosik1 = new Stos();
stosik1.push(2);
stosik1.push(4);
stosik1.show();
stosik1.push(6);
stosik1.show();
stosik1.pop();
stosik1.show();
Console.WriteLine(stosik1.getcount());
Console.WriteLine(stosik1.isfull());
stosik1.pop();
stosik1.pop();
stosik1.show();
Console.WriteLine(stosik1.isempty());

```

Wynik działania kodu:

```
Konsola debugowania programu Microsoft Visual Studio

Filip Rzeplia Grupa P2
Kolejka: 1 5
Kolejka: 1 5 7
Kolejka: 1 5 7 9
Kolejka: 5 7 9
3
Kolejka: 5 7 9 4 5
5
Kolejka:
0
True
False
Kolejka: 2 2 2 2 2 2 2 2 2 2
10
True
Stos: 2 4
Stos: 2 4 6
Stos: 2 4
2
False
Stos:
True

X:\NSI\NSI_01\NSI_1_RF\NSI_1_RF\bin\Debug\netcoreapp3.1\NSI_1_RF.exe (proces 19560) zakończono z kodem 0.
Aby automatycznie zamknąć konsolę po zatrzymaniu debugowania, włącz opcję Narzędzia -> Opcje -> Debugowanie -> Automatycznie zamknij konsolę po zatrzymaniu debugowania.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

### Zadanie 3

Wykonaj implementację, która w momencie tworzenia obiektu pozwala zdecydować o ilości miejsca w buforze oraz typie elementów (typ generyczny), na którym operuje implementowany kontener. (na ocenę bdb).

Decydowanie o ilości miejsca w buforze

Typ genetyczny – tworzymy przez dodanie <T> przy klasie

```
abstract class Container<T>
{
    protected int pointer = -1;
    protected T[] buffer = new T[10];
    protected int size = 10;
    public abstract void show();
    public abstract T pop();
    public abstract void push(T value);
    public Container(){ }
    public Container(int size_)
    {
        if (size > 0)
        {
            buffer = new T[size_];
            size = size_;
        }
    }
}
```

Działanie kodu:

```
Konsola debugowania programu Microsoft Visual Studio
Filip Rzepiela Grupa P2
Kolejka
1 5
Kolejka
1 5 7
Kolejka
1 5 7 9
Kolejka
5 7 9
3
Kolejka
5 7 9 4
4
Kolejka
0
True
False
Kolejka
2 2 2 2
4
True
Stos
2 4
Stos
2 4 6
Stos
2 4
2
False
Stos
True

X:\NSI\NSI_01\NSI_1_RF\NSI_1_RF\bin\Debug\netcoreapp3.1\NSI_1_RF.exe (proces 8440) zakończono z kodem 0.
Aby automatycznie zamknąć konsolę po zatrzymaniu debugowania, włącz opcję Narzędzia -> Opcje -> Debugowanie -> Automatycznie zamknij konsolę po zatrzymaniu debugowania.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```