Algoritmi in podatkovne strukture 1

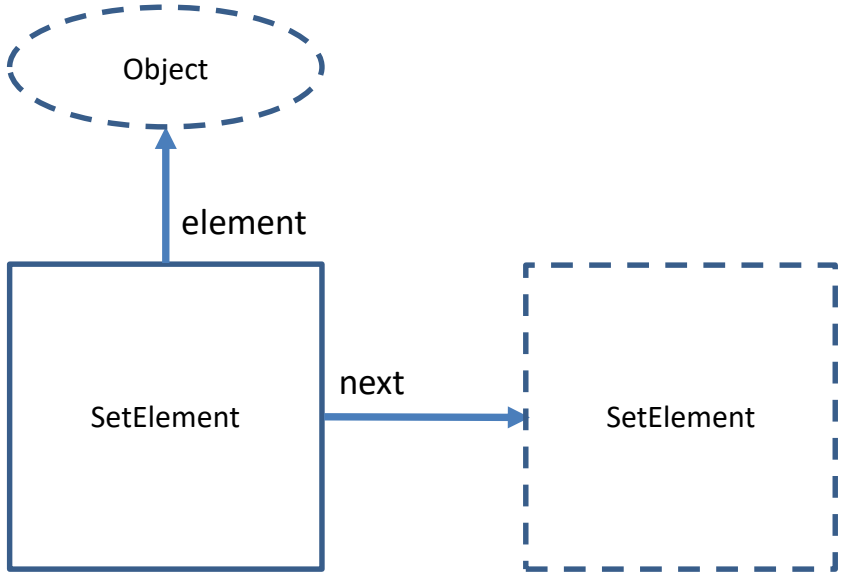
Laboratorijske vaje

Množica

MNOŽICA

```
class SetElement
{
    Object element;
    SetElement next;
    …
}
```

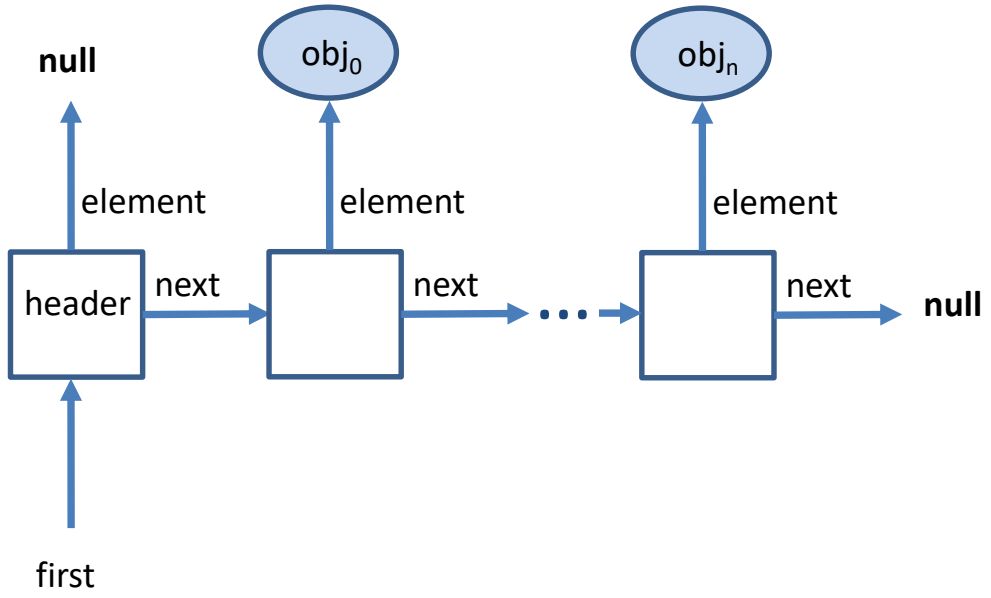
```
class Set
{
    SetElement first;
    …
}
```



MNOŽICA

Osnovne operacije:

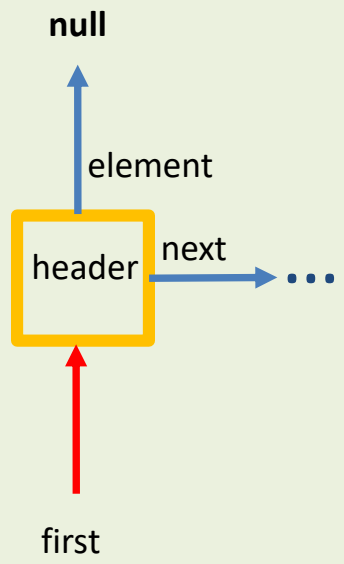
- insert
- locate
- delete



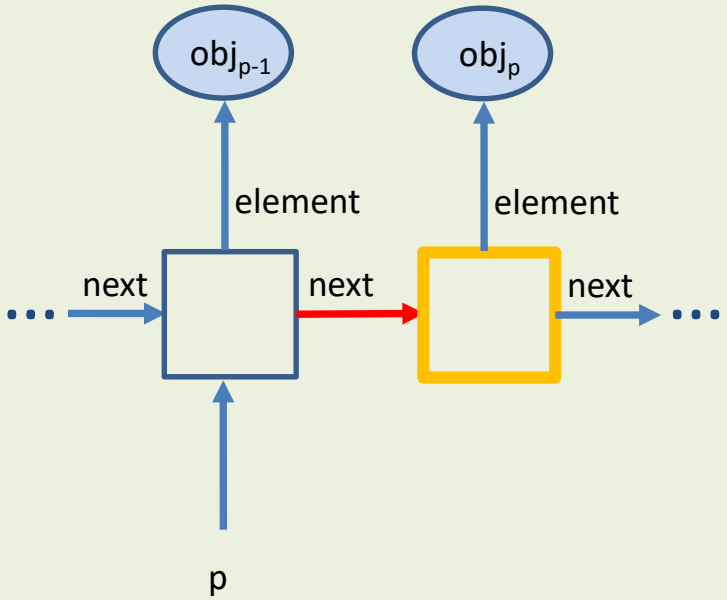
MNOŽICA

Že implementirano…

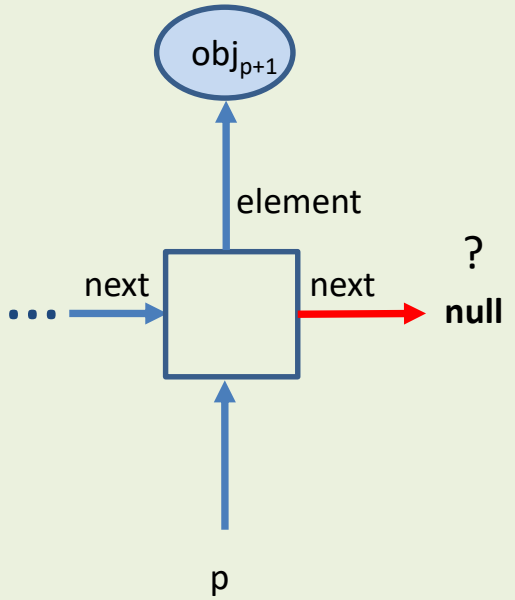
```
public SetElement first()
{
    return first;
}
```




```
public SetElement next(SetElement p)
{
    return p.next;
}
```



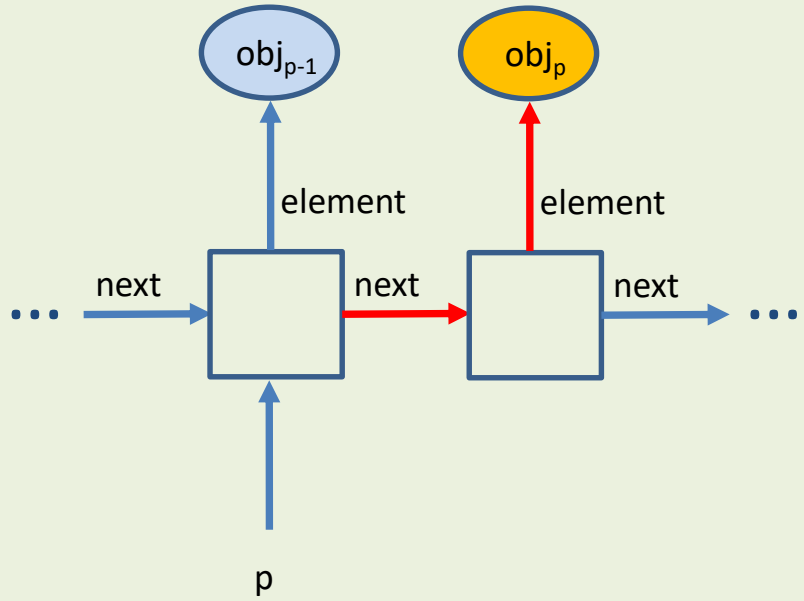

```
public boolean overEnd(SetElement p)
{
    if (p.next == null)
        return true;
    else
        return false;
}
```



MNOŽICA

Že implementirano…

```
public Object retrieve(SetElement p)
{
    return p.next.element;
}
```



MNOŽICA

Že implementirano…

```
public void print()
{
    System.out.print("{");

    for (SetElement iter = first(); !overEnd(iter); iter = next(iter))
    {
        System.out.print(retrieve(iter));

        if (!overEnd(next(iter)))
            System.out.print(", ");
    }

    System.out.println("}");
}
```

NALOGE

Implementirajte naslednje metode v razredu Set:

- `void insert(Object obj)` – doda element *obj* v množico (brez podvajanja!)
- `SetElement locate(Object obj)` – vrne položaj iskanega elementa *obj*
- `void delete(SetElement pos)` - odstrani element na poziciji *pos*
- `void union(Set a)` - brez podvajanja doda vse elemente množice *a*
- `void intersection(Set a)` - odstrani vse elemente, ki se ne nahajajo tudi v množici *a*

NALOGE

Primer:

```
Set x = new Set();
x.insert(1);
x.insert(2);
x.insert(3);

Set y = new Set();
y.insert(5);
y.insert(2);

x.union(y);         // množica x sedaj vsebuje elemente 1,2,3 in 5

Set z = new Set();
z.insert(4);
z.insert(2);
z.insert(3);

x.intersection(z); // v množici x sta sedaj samo 2 in 3
```

NALOGE

Implementirajte naslednje metode v razredu GlavniProgram:

- `void crke(String stavek)` – najprej izpiše vse črke, ki nastopajo v stavku, nato še vse črke, ki nastopajo v vsaki besedi stavka

```
crke("Abstraktni podatkovni tip")
```

M_1 M_2 M_3

- `Set createPowerSet(Set s)` – generira potenčno množico podane množice *s*

POTENČNA MNOŽICA

Primer: generiraj potenčno množico podane množice S = {a, b, c}.

Pristop z binarnimi nizi dolžine n = m(S)

	abc	Podmnožica
0	000	{}
1	001	{c}
2	010	{b}
3	011	{b,c}
4	100	{a}
5	101	{a,c}
6	110	{a,b}
7	111	{a,b,c}

$\mathscr{P}_{\{a,b,c\}}$ = {{},{c}, {b}, {b,c}, {a}, {a,c}, {a,b}, {a,b,c}}

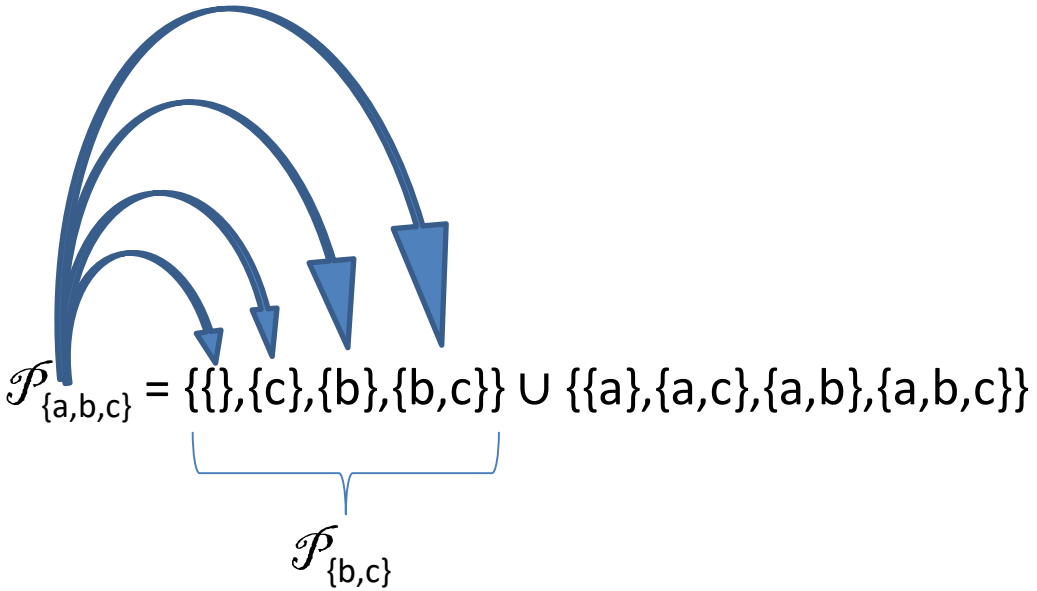
POTENČNA MNOŽICA

Primer: generiraj potenčno množico podane množice S = {a, b, c}.

Rekurzivna definicija potenčne množice:

$\mathcal{P}_{\{\}} = \{\{\}\}$

$\mathcal{P}_{\{a,Ostali\ elementi\}} = \mathcal{P}_{\{Ostali\ elementi\}} \cup \{t \cup \{a\} : t \in \mathcal{P}_{\{Ostali\ elementi\}}\}$



POTENČNA MNOŽICA

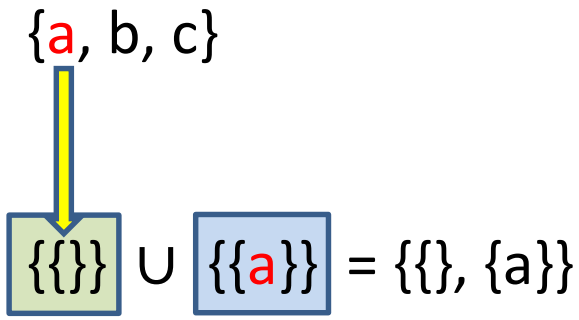
Primer: generiraj potenčno množico podane množice S = {a, b, c}.

{a, b, c}

{{}}

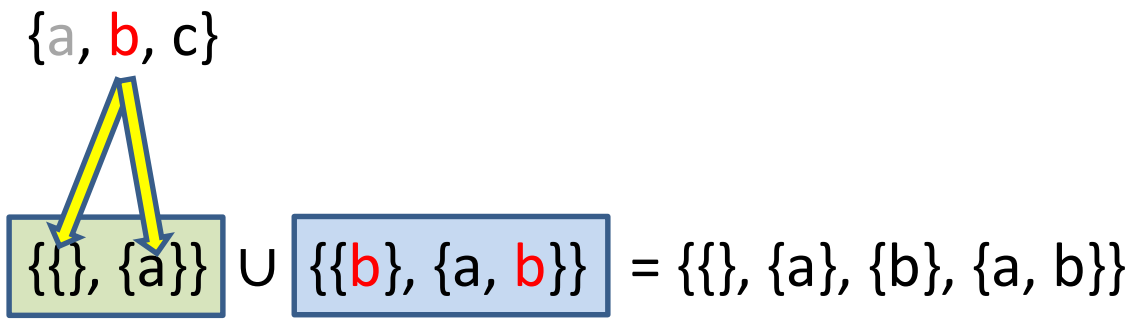
POTENČNA MNOŽICA

Primer: generiraj potenčno množico podane množice S = {a, b, c}.



POTENČNA MNOŽICA

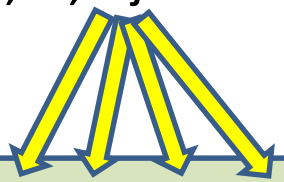
Primer: generiraj potenčno množico podane množice S = {a, b, c}.



POTENČNA MNOŽICA

Primer: generiraj potenčno množico podane množice S = {a, b, c}.

{a, b, c}



{{}, {a}, {b}, {a, b}} ∪ {{c}, {a, c}, {b, c}, {a, b, c}} = {{}, {a}, {b}, {a, b}, {c}, {a, c}, {b, c}, {a, b, c}}