LAPORAN PRAKTIKUM PBO
Java Generic Class

Disusun oleh :
Muhamad Rafli Nur Ikhsan
201511048
D-3 Teknik Informatika 2B

Jurusan Teknik Komputer dan Informatika

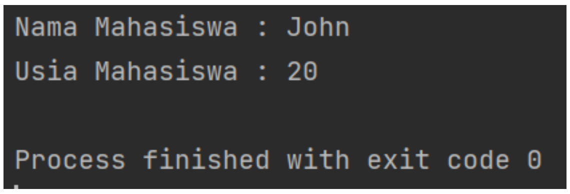
Program studi D3 Teknik Informatika

Politeknik Negeri Bandung

1. Kapan penggunaan generic?
   Saat membuat class/method yang bertujuan membangun objek yang sama
   namun dengan tipe data yang berbeda,
2. Apa keuntungannya jika mengimplementasikan generic programming baik pada class, method, interface dll?
   Dengan kita mengimplementasikan generic programming, ini memungkinkan
   untuk sebuah single class untuk bekerja dengan pilihan tipe data yang lebih beragam.
   Hal ini merupakan yang terbaik daripada harus menggunakan casting atau membuat
   class baru dengan tipe data yang berbeda.
3. Berikan contoh konkrit kasus lain dari ke 5 contoh generic (Selain file yang dilampirkan) yang bisa anda implementasikan . contoh konkrit dibolehkan 1 kasus namun 5 generic yang diminta tersedia.
   - Generic Class
     - Source Code

```
public class GenericClass<T>{

    private T t;

    public GenericClass(T t) {
        this.t = t;
    }

    public T get(){
        return this.t;
    }
    public void set(T t1){
        this.t=t1;
    }
    public static void main(String[] args) {
        GenericClass<String> Nama = new
GenericClass<String>("John");

        GenericClass<Integer> Usia = new
GenericClass<Integer>(20);

        String  nama = Nama.get();
        Integer usia = Usia.get();

        System.out.println("Nama  : " + nama);
        System.out.println("Usia  : " + usia);
    }

}
```

     - SS akhir program

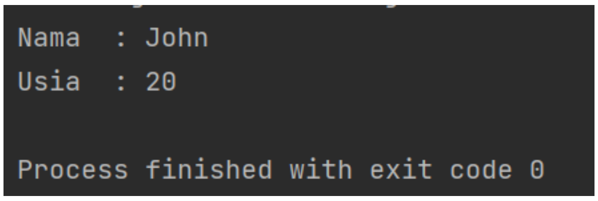


```
Nama Mahasiswa : John
Usia Mahasiswa : 20

Process finished with exit code 0
```

   - Generic Method
     - Source Code

```
public class GenericsType<T> {
    private T t;
    public T get(){
        return this.t;
    }
    public void set(T t1){
        this.t=t1;
    }
    public static void main(String args[]){
        GenericsType<String> type = new GenericsType<>();
        type.set("John"); //valid
        GenericsType type1 = new GenericsType(); //raw
type
        type1.set(20); //valid and autoboxing support
        System.out.println("Nama  : " + type.get());
        System.out.println("Usia  : " + type1.get());
    }
}
public class GenMethod {
        public static <T> boolean isEqual(GenericsType<T>
g1, GenericsType<T> g2){
            return g1.get().equals(g2.get());
        }
        public static void main(String args[]){
            GenericsType<String> g1 = new
GenericsType<>();
            g1.set("Java");
            GenericsType<String> g2 = new
GenericsType<>();
            g2.set("Java");
            boolean isEqual =
GenMethod.<String>isEqual(g1, g2);

            isEqual = GenMethod.isEqual(g1, g2);

        }
    }
```

- SS akhir program

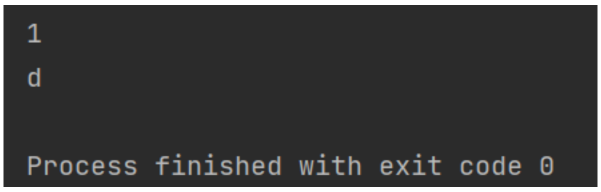


```
Nama  : John
Usia  : 20

Process finished with exit code 0
```

- Generic Interface
  - Source Code

```
class GenClass<T extends Comparable<T>> implements
GenInter<T> {
    T[] vals;
    GenClass(T[] o)  {
        vals = o;
    }
    public T min()  {
        T v = vals[0];
        for (int i = 1; i < vals.length; i++)  {
            if  (vals[i].compareTo(v)  < 0)  {
                v = vals[i];
            }
        }
        return v;
    }
}
interface GenInter<T extends Comparable<T>> {
    T min(); /* w w w .java2 s . co m*/
}
public class Main {
    public static void main(String args[]) {
        Integer inums[] = { 9, 1, 2, 8, 4 };
        Character chs[] = { 'r', 'i', 'j', 'd' };
        GenClass<Integer> a = new
GenClass<Integer>(inums);
        GenClass<Character> b = new
GenClass<Character>(chs);
        System.out.println(a.min());
        System.out.println(b.min());
    }
}
```

- SS akhir program

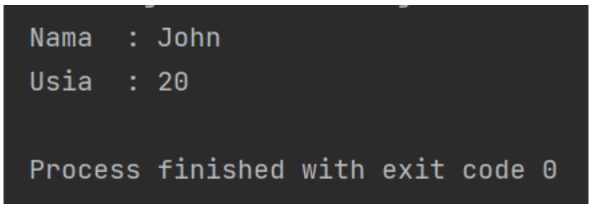


```
1
d

Process finished with exit code 0
```

- Generic Bounded
  - Source Code

```
public class BoundedTypeParameter<T> {
    private T BTP;

    public BoundedTypeParameter(T BTP){
        this.BTP = BTP;
    }

    public T getBTP(){
        return BTP;
    }

    public void setBTP(T BTP){
        this.BTP = BTP;
    }


}
public class Bounded {

    public static void main(String[] args) {
        BoundedTypeParameter<String> Nama = new
BoundedTypeParameter<>("John");
        BoundedTypeParameter<Integer> Umur = new
BoundedTypeParameter<>(20);

        String nama =Nama.getBTP();
        Integer umur = Umur.getBTP();
        System.out.println("Nama   : " + nama);
        System.out.println("Usia   : " + umur);

    }

}
```

  - SS akhir program

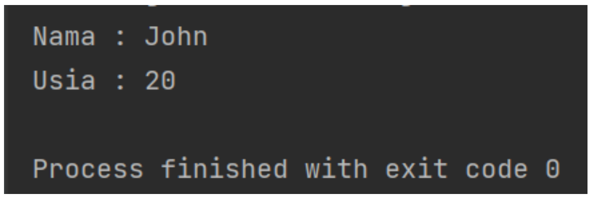


```
Nama  : John
Usia  : 20

Process finished with exit code 0
```

- Generic Wildcard
  - Source Code

```java
public class Wildcard <T>{

    private T Wildcard;

    public Wildcard(T Wildcard){
        this.Wildcard = Wildcard;
    }

    public T getWildcard(){
        return Wildcard;
    }

    public void setWildcard(T Wildcard){
        this.Wildcard = Wildcard;
    }
}
public class Main {

    public static void main(String[] args)  {
        printValue(new Wildcard<>("Nama : " + "John"));
        printValue(new Wildcard<>("Usia : "+ 20));

    }
    public static void printValue(Wildcard<?> Wildcard)  {
        System.out.println(Wildcard.getWildcard());
    }
}
```

  - SS akhir program



```
Nama : John
Usia : 20

Process finished with exit code 0
```