Java Programming I

Session 9

Streams

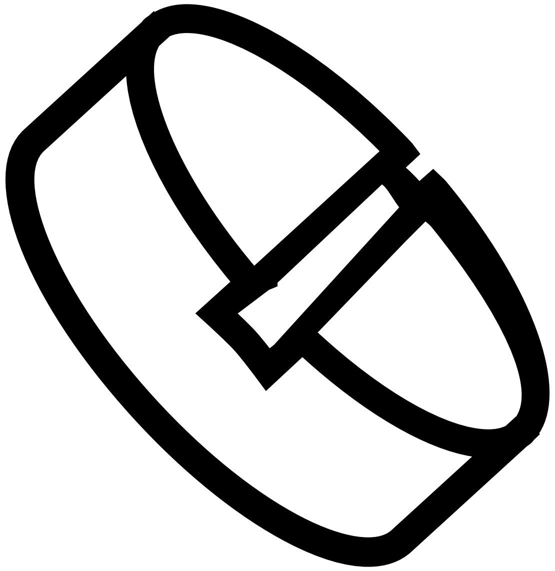
Juan Carlos Moreno - UCLA Ex

Agenda

- Generic Fundamentals
- Wildcards and bounds
- Methods
- Constructors
- Interfaces
- Restrictions

Fundamentals

write once, use many times



```
public class MyGeneric<T>
```

Generic considerations

Things to keep in mind

- Generics don’t work with primitives (int, float, etc)
- They only work with the type they are created
- Generics can have a parameter list <T, V…>

Wildcards and bounds

Read/Write while there’s something...

Bounding can limit types accepted

<T extends *superclass*>

Wildcards allow using undeclared types

Parameter<?>

Bounded wildcards

Parameter<? extends *superclass*>

Generic Methods

Read/Write while there’s something...

```
public <T extends Number> stddev(T[] args){
    // Do Something
    return args[0];
}
```

Generic Interface

the framework

```
interface Recipe<T extends Ingredient>{
    boolean mixIngredients();
}
```

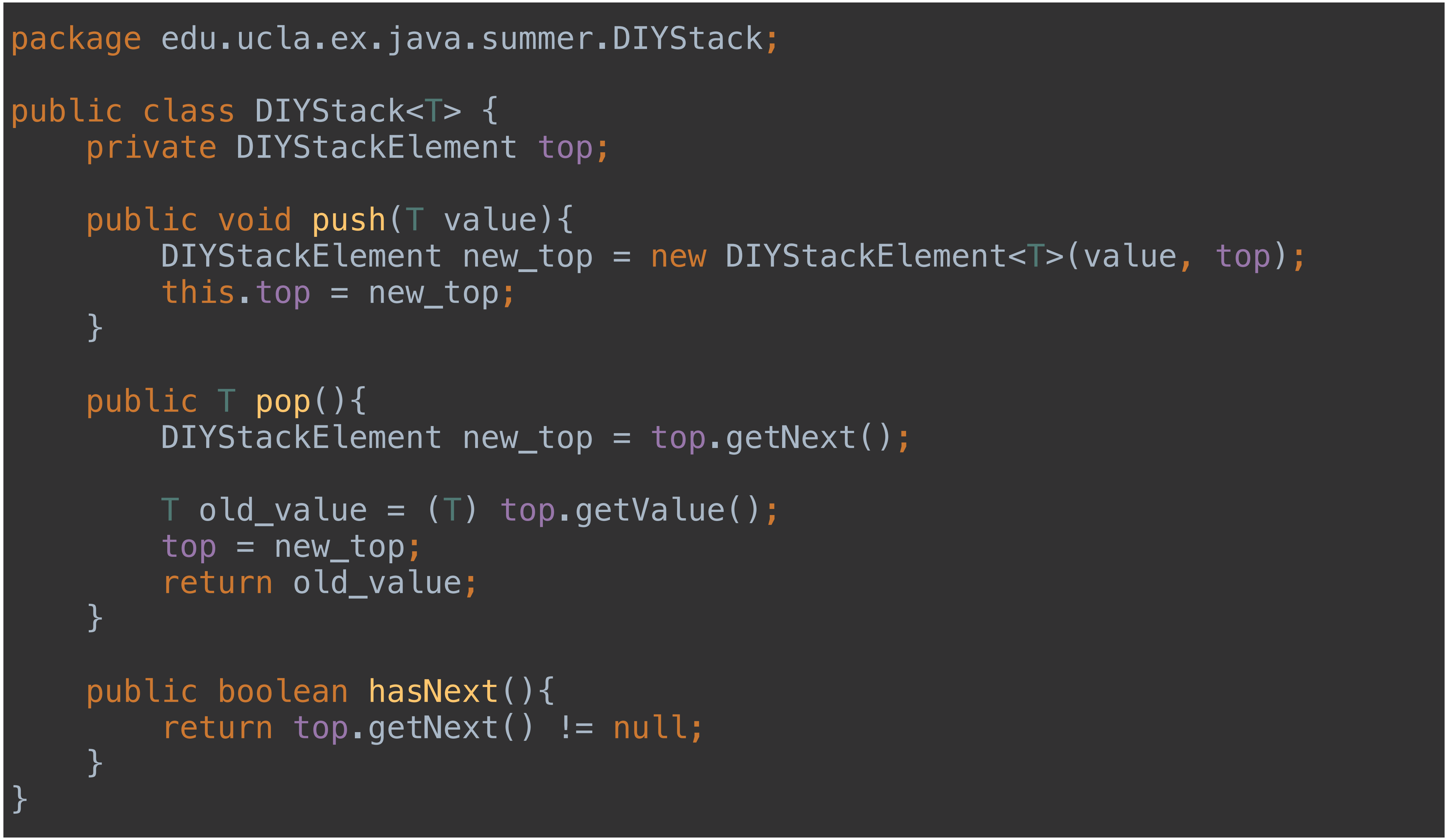
Generic Example

An old friend

```java
package edu.ucla.ex.java.summer.DIYStack;

public class DIYStackElement<T> {
    private DIYStackElement next;
    private T value;

    public DIYStackElement(T _value, DIYStackElement _next) {
        value = _value;
        next = _next;
    }

    public DIYStackElement getNext(){
        return next;
    }

    public T getValue(){
        return value;
    }

}
```

Generic Example

old friend extended



```java
package edu.ucla.ex.java.summer.DIYStack;

public class DIYStack<T> {
    private DIYStackElement top;

    public void push(T value){
        DIYStackElement new_top = new DIYStackElement<T>(value, top);
        this.top = new_top;
    }

    public T pop(){
        DIYStackElement new_top = top.getNext();

        T old_value = (T) top.getValue();
        top = new_top;
        return old_value;
    }

    public boolean hasNext(){
        return top.getNext() != null;
    }
}
```

Restrictions

Keep these in mind

- Ambiguity Errors
- Type params can’t be instantiated
- No static members
- Cannot extend from throwable
- Overloading can be ambiguous

Exercise

Queue

- First in, First out
- Queue and deQueue