

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

Generics

The INFDEV team

Hogeschool Rotterdam
Rotterdam, Netherlands

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

Introduction

Introduction

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

Lecture topics

- Arrays as a simple generic data type
- Class generators: generics
- Interfaces and generics
- Generic lists: a concrete example
- Lambda

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

Arrays as a simple generic data type

Arrays as a simple generic data type

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

Introduction

- A very common necessity when programming is storing multiple values in a variable
- There actually is a built-in datatype in most programming languages to do so
- This datatype is called **array**

Arrays as a simple generic data type

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

Introduction

- An array is declared with the type of the element, followed by square brackets
- The array is then initialized by specifying the number of elements it can store
- The elements are then written and accessed given their position in the array
- The array cannot change size: reading or writing an elements out of bounds gives an error

Arrays as a simple generic data type

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

An array is declared with the type of the element, followed by square brackets

```
1 int [] x;
```

Arrays as a simple generic data type

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

Which in Java then becomes:

```
1 int[] x;
```

Arrays as a simple generic data type

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

The array is then initialized by specifying the number of elements it can store

```
1 int [] x = new int[10];
```

Arrays as a simple generic data type

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

Which in Java then becomes:

```
1 int [] x = new int [10];
```

Arrays as a simple generic data type

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

The elements are then written and accessed given their position in the array

```
1 int[] x = new int[10];  
2 x[5] = 100;  
3 Console.WriteLine(x[5]);
```

Arrays as a simple generic data type

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

Which in Java then becomes:

```
1 int[] x = new int[10];  
2 x[5] = 100;  
3 System.out.println(x[5]);
```

Arrays as a simple generic data type

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

```
1 int [] x = new int [10];  
2 x[5] = 100;  
3 Console.WriteLine(x[5]);
```

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

Stack:

PC
1

Arrays as a simple generic data type

Generics

The INFDEV
team

Introduction

```
1 int[] x = new int[10];  
2 x[5] = 100;  
3 Console.WriteLine(x[5]);
```

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

Stack:

PC
1

Heap:

1
[; ; ; ; ; ; ; ;]

Arrays as a simple generic data type

Generics

The INFDEV
team

Introduction

```
1 int[] x = new int[10];  
2 x[5] = 100;  
3 Console.WriteLine(x[5]);
```

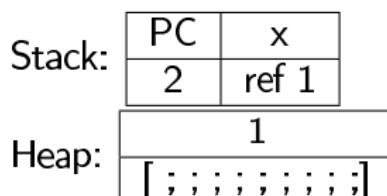
Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion



Arrays as a simple generic data type

Generics

The INFDEV
team

Introduction

```
1 int[] x = new int[10];  
2 x[5] = 100;  
3 Console.WriteLine(x[5]);
```

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

Stack:

PC	x
3	ref 1

Heap:

1
[; ; ; ; ; 100 ; ; ; ;]

Arrays as a simple generic data type

Generics

The INFDEV
team

Introduction

```
1 int[] x = new int[10];  
2 x[5] = 100;  
3 Console.WriteLine(x[5]);
```

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

Stack:

PC	x
4	ref 1

Heap:

1
[; ; ; ; 100 ; ; ;]

Output:

100

Arrays as a simple generic data type

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

The array cannot change size: reading or writing an elements out of bounds gives an error

The program below would just crash at runtime with an *array out of bounds error*

```
1 int [] x = new int [10];  
2 x[15] = 100;  
3 Console.WriteLine(x[15]);
```

Arrays as a simple generic data type

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

Which in Java then becomes:

```
1 int[] x = new int[10];  
2 x[15] = 100;  
3 System.out.println(x[15]);
```

Arrays as a simple generic data type

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

Arrays of various types?

- Arrays can come in all sorts of types
- int[], float[], bool[], string[], Car[], ...

Arrays as a simple generic data type

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

So what is common to all arrays, independently of their specific content?

Arrays as a simple generic data type

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

So what is common to all arrays, independently of their specific content?

- An array $T[]$ contains a series of elements **of any type T**
- It is initialized with `new T[n]`, where n is the number of stored elements
- We access the i -th element of array a of type $T[]$ with $a[i]$; $a[i]$ has type T
- We set the i -th element of array a of type $T[]$ with $a[i] = e$; e has type T

Arrays as a simple generic data type

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

Arrays of various types?

So it makes perfect sense to speak about arrays in terms which are **generic** with respect to the type of the elements!

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

Class generators: generics

Class generators: generics

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

Introduction

- We can define classes that follow the same philosophy just explained for arrays
- These classes only specify a structure, but are independent of the type of their content
- These classes are called **generic classes**

Class generators: generics

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

Suppose we wish to define a class that stores two elements together.

- It can be useful in many places when we want to couple two things together
- Return two values from a function
- Store a list of relationships
- ...

Class generators: generics

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

Storing two elements together should be independent of their type. We do not want to define a new version of the class for each possible combination, such as:

- int and bool
- float and float
- bool and string
- Car and int
- Person and Dog
- Man and Woman
- Woman and Man
- Woman and Woman
- Man and Man
- ...

Class generators: generics

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

```
1 class Pair<T, U> {
2     private T x;
3     private U y;
4     public Pair(T x, U y) {
5         this.x = x;
6         this.y = y;
7     }
8     public T First() {
9         return this.x;
10    }
11    public U Second() {
12        return this.y;
13    }
14 }
```

Class generators: generics

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

Which in Java then becomes:

```
1 class Pair<T, U> {
2     private T x;
3     private U y;
4     public Pair(T x, U y) {
5         this.x = x;
6         this.y = y;
7     }
8     public T First() {
9         return this.x;
10    }
11    public U Second() {
12        return this.y;
13    }
14 }
```

Class generators: generics

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

We can then use this class by specifying what the types of its fields are concretely:

```
1  class Pair<T, U> {
2      private T x;
3      private U y;
4      public Pair(T x,U y) {
5          this.x = x;
6          this.y = y;
7      }
8      public T First() {
9          return this.x;
10     }
11     public U Second() {
12         return this.y;
13     }
14 }
15 Pair<int, bool> p = new Pair<int, bool>(10,true);
16 Console.WriteLine(p.First());
17 Console.WriteLine(p.Second());
```

Class generators: generics

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

Which in Java then becomes:

```
1 class Pair<T, U> {  
2     private T x;  
3     private U y;  
4     public Pair(T x, U y) {  
5         this.x = x;  
6         this.y = y;  
7     }  
8     public T First() {  
9         return this.x;  
10    }  
11    public U Second() {  
12        return this.y;  
13    }  
14}  
15 Pair<Integer, Boolean> p = new Pair<Integer, Boolean>(10, true);  
16 System.out.println(p.First());  
17 System.out.println(p.Second());
```

Class generators: generics

Generics

The INFDEV team

Introduction

Arrays as a simple generic data type

Class generators: generics

Interfaces and generics

Generic lists: a concrete example

Lambda functions

Conclusion

```
1 class Pair<T, U> {
2     private T x;
3     private U y;
4     public Pair(T x, U y) {
5         this.x = x;
6         this.y = y;
7     }
8     public T First() {
9         return this.x;
10    }
11    public U Second() {
12        return this.y;
13    }
14 }
15 Pair<int, bool> p = new Pair<int, bool>(10, true);
16 Console.WriteLine(p.First());
17 Console.WriteLine(p.Second());
```

Stack:

PC
1

Class generators: generics

Generics

The INFDEV team

Introduction

Arrays as a simple generic data type

Class generators: generics

Interfaces and generics

Generic lists: a concrete example

Lambda functions

Conclusion

```
1 class Pair<T, U> {
2     private T x;
3     private U y;
4     public Pair(T x, U y) {
5         this.x = x;
6         this.y = y;
7     }
8     public T First() {
9         return this.x;
10    }
11    public U Second() {
12        return this.y;
13    }
14 }
15 Pair<int, bool> p = new Pair<int, bool>(10, true);
16 Console.WriteLine(p.First());
17 Console.WriteLine(p.Second());
```

Stack:

PC
15

Class generators: generics

Generics

The INFDEV team

Introduction

Arrays as a simple generic data type

Class generators: generics

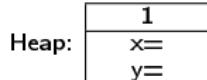
Interfaces and generics

Generic lists: a concrete example

Lambda functions

Conclusion

```
1 class Pair<T, U> {
2     private T x;
3     private U y;
4     public Pair(T x, U y) {
5         this.x = x;
6         this.y = y;
7     }
8     public T First() {
9         return this.x;
10    }
11    public U Second() {
12        return this.y;
13    }
14 }
15 Pair<int, bool> p = new Pair<int, bool>(10, true);
16 Console.WriteLine(p.First());
17 Console.WriteLine(p.Second());
```



Class generators: generics

Generics

The INFDEV team

Introduction

Arrays as a simple generic data type

Class generators: generics

Interfaces and generics

Generic lists: a concrete example

Lambda functions

Conclusion

```
1 class Pair<T, U> {
2     private T x;
3     private U y;
4     public Pair(T x, U y) {
5         this.x = x;
6         this.y = y;
7     }
8     public T First() {
9         return this.x;
10    }
11    public U Second() {
12        return this.y;
13    }
14 }
15 Pair<int, bool> p = new Pair<int, bool>(10, true);
16 Console.WriteLine(p.First());
17 Console.WriteLine(p.Second());
```

Stack:	PC	...	PC	ret	this	x	y
	15	...	5	null	ref 1	10	true

Heap:	1
x=	
y=	

Class generators: generics

Generics

The INFDEV team

Introduction

Arrays as a simple generic data type

Class generators: generics

Interfaces and generics

Generic lists: a concrete example

Lambda functions

Conclusion

```
1 class Pair<T, U> {
2     private T x;
3     private U y;
4     public Pair(T x, U y) {
5         this.x = x;
6         this.y = y;
7     }
8     public T First() {
9         return this.x;
10    }
11    public U Second() {
12        return this.y;
13    }
14 }
15 Pair<int, bool> p = new Pair<int, bool>(10, true);
16 Console.WriteLine(p.First());
17 Console.WriteLine(p.Second());
```

Stack:	PC	...	PC	ret	this	x	y
	15	...	6	null	ref 1	10	true

Heap:	1
x=10	
y=	

Class generators: generics

Generics

The INFDEV team

Introduction

Arrays as a simple generic data type

Class generators: generics

Interfaces and generics

Generic lists: a concrete example

Lambda functions

Conclusion

```
1 class Pair<T, U> {
2     private T x;
3     private U y;
4     public Pair(T x, U y) {
5         this.x = x;
6         this.y = y;
7     }
8     public T First() {
9         return this.x;
10    }
11    public U Second() {
12        return this.y;
13    }
14 }
15 Pair<int, bool> p = new Pair<int, bool>(10, true);
16 Console.WriteLine(p.First());
17 Console.WriteLine(p.Second());
```

Stack:	PC	...	PC	ret	this	x	y
	15	...	6	null	ref 1	10	true

Heap:	1
	x=10 y=true

Class generators: generics

Generics

The INFDEV team

Introduction

Arrays as a simple generic data type

Class generators: generics

Interfaces and generics

Generic lists: a concrete example

Lambda functions

Conclusion

```
1 class Pair<T, U> {
2     private T x;
3     private U y;
4     public Pair(T x, U y) {
5         this.x = x;
6         this.y = y;
7     }
8     public T First() {
9         return this.x;
10    }
11    public U Second() {
12        return this.y;
13    }
14 }
15 Pair<int, bool> p = new Pair<int, bool>(10, true);
16 Console.WriteLine(p.First());
17 Console.WriteLine(p.Second());
```

Stack:	PC	...	PC	ret
	15	...	6	ref 1

Heap:	1
	x=10 y=true

Class generators: generics

Generics

The INFDEV team

Introduction

Arrays as a simple generic data type

Class generators: generics

Interfaces and generics

Generic lists: a concrete example

Lambda functions

Conclusion

```
1 class Pair<T, U> {
2     private T x;
3     private U y;
4     public Pair(T x, U y) {
5         this.x = x;
6         this.y = y;
7     }
8     public T First() {
9         return this.x;
10    }
11    public U Second() {
12        return this.y;
13    }
14 }
15 Pair<int, bool> p = new Pair<int, bool>(10, true);
16 Console.WriteLine(p.First());
17 Console.WriteLine(p.Second());
```

Stack:	PC	p
	16	ref 1

Heap:	1
	x=10 y=true

Class generators: generics

Generics

The INFDEV team

Introduction

Arrays as a simple generic data type

Class generators: generics

Interfaces and generics

Generic lists: a concrete example

Lambda functions

Conclusion

```
1 class Pair<T, U> {
2     private T x;
3     private U y;
4     public Pair(T x, U y) {
5         this.x = x;
6         this.y = y;
7     }
8     public T First() {
9         return this.x;
10    }
11    public U Second() {
12        return this.y;
13    }
14 }
15 Pair<int, bool> p = new Pair<int, bool>(10, true);
16 Console.WriteLine(p.First());
17 Console.WriteLine(p.Second());
```

Stack:	PC	...	PC	ret	this
	16	...	9	null	ref 1

Heap:	1
	x=10 y=true

Class generators: generics

Generics

The INFDEV team

Introduction

Arrays as a simple generic data type

Class generators: generics

Interfaces and generics

Generic lists: a concrete example

Lambda functions

Conclusion

```
1 class Pair<T, U> {
2     private T x;
3     private U y;
4     public Pair(T x, U y) {
5         this.x = x;
6         this.y = y;
7     }
8     public T First() {
9         return this.x;
10    }
11    public U Second() {
12        return this.y;
13    }
14 }
15 Pair<int, bool> p = new Pair<int, bool>(10, true);
16 Console.WriteLine(p.First());
17 Console.WriteLine(p.Second());
```

Stack:	PC	...	PC	ret
	16	...	9	10

Heap:	1
	x=10 y=true

Class generators: generics

Generics

The INFDEV team

Introduction

Arrays as a simple generic data type

Class generators: generics

Interfaces and generics

Generic lists: a concrete example

Lambda functions

Conclusion

```
1 class Pair<T, U> {
2     private T x;
3     private U y;
4     public Pair(T x, U y) {
5         this.x = x;
6         this.y = y;
7     }
8     public T First() {
9         return this.x;
10    }
11    public U Second() {
12        return this.y;
13    }
14 }
15 Pair<int, bool> p = new Pair<int, bool>(10, true);
16 Console.WriteLine(p.First());
17 Console.WriteLine(p.Second());
```

Stack:

PC	p
17	ref 1

Heap:

1
x=10
y=true

Output: 10

Class generators: generics

Generics

The INFDEV team

Introduction

Arrays as a simple generic data type

Class generators: generics

Interfaces and generics

Generic lists: a concrete example

Lambda functions

Conclusion

```
1 class Pair<T, U> {
2     private T x;
3     private U y;
4     public Pair(T x, U y) {
5         this.x = x;
6         this.y = y;
7     }
8     public T First() {
9         return this.x;
10    }
11    public U Second() {
12        return this.y;
13    }
14 }
15 Pair<int, bool> p = new Pair<int, bool>(10, true);
16 Console.WriteLine(p.First());
17 Console.WriteLine(p.Second());
```

Stack:	PC	...	PC	ret	this
	17	...	12	null	ref 1

Heap:	1
x=10	
y=true	

Output: 10

Class generators: generics

Generics

The INFDEV team

Introduction

Arrays as a simple generic data type

Class generators: generics

Interfaces and generics

Generic lists: a concrete example

Lambda functions

Conclusion

```
1 class Pair<T, U> {
2     private T x;
3     private U y;
4     public Pair(T x, U y) {
5         this.x = x;
6         this.y = y;
7     }
8     public T First() {
9         return this.x;
10    }
11    public U Second() {
12        return this.y;
13    }
14 }
15 Pair<int, bool> p = new Pair<int, bool>(10, true);
16 Console.WriteLine(p.First());
17 Console.WriteLine(p.Second());
```

Stack:	PC	...	PC	ret
	17	...	12	true

Heap:	1
x=10	
y=true	

Output: 10

Class generators: generics

Generics

The INFDEV team

Introduction

Arrays as a simple generic data type

Class generators: generics

Interfaces and generics

Generic lists: a concrete example

Lambda functions

Conclusion

```
1 class Pair<T, U> {
2     private T x;
3     private U y;
4     public Pair(T x, U y) {
5         this.x = x;
6         this.y = y;
7     }
8     public T First() {
9         return this.x;
10    }
11    public U Second() {
12        return this.y;
13    }
14 }
15 Pair<int, bool> p = new Pair<int, bool>(10, true);
16 Console.WriteLine(p.First());
17 Console.WriteLine(p.Second());
```

Stack:	PC	p
	18	ref 1

Heap:	1
x=10	
y=true	

Output: 10 true

Class generators: generics

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

An example: arbitrary pairs

- In Java, generic arguments cannot be all those primitive types with non-reference values that sit directly on the stack
- This means that we cannot write `Pair<int,int>` in Java
- The standard library contains **reference versions** of those types, starting with a capital letter, such as `Integer`, etc.
- Those types are like the primitive types, but their values are references that point to the actual value on the heap
- We can then write `Pair<Integer,Integer>`

Class generators: generics

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

The types of the fields can change, but the class implementation always remains the same:

```
1 Pair<int, bool> p = new Pair<int, bool>(10,true);
2 Pair<float, bool> p = new Pair<float, bool>(10,false);
3 Pair<bool, bool> p = new Pair<bool, bool>(false,true);
4 Pair<string, int> p = new Pair<string, int>("First item",5);
5 ...
```

Class generators: generics

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

Which in Java then becomes:

```
1 Pair<Integer, Boolean> p = new Pair<Integer, Boolean>(10,true);
2 Pair<Float, Boolean> p = new Pair<Float, Boolean>(10,false);
3 Pair<Boolean, Boolean> p = new Pair<Boolean, Boolean>(false,true);
4 Pair<String, Integer> p = new Pair<String, Integer>("First item",5);
5 ...
```

Class generators: generics

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

Typechecking of generic classes

- Typecheking is simply a form of **substitution**
- When the class is instantiated with types as arguments, a new version of the class is created
- The created class has the concrete versions of these parameters in it

Class generators: generics

Generics

The INFDEV team

Introduction

Arrays as a simple generic data type

Class generators: generics

Interfaces and generics

Generic lists: a concrete example

Lambda functions

Conclusion

```
1 class Pair<T, U> {
2     private T x;
3     private U y;
4     public Pair(T x,U y) {
5         this.x = x;
6         this.y = y;
7     }
8     public T First() {
9         return this.x;
10    }
11    public U Second() {
12        return this.y;
13    }
14 }
15 Pair<int, bool> p = new Pair<int, bool>(10,true);
```

Declarations:	PC
	1

Class generators: generics

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

```
1 class Pair<T, U> {
2     private T x;
3     private U y;
4     public Pair(T x, U y) {
5         this.x = x;
6         this.y = y;
7     }
8     public T First() {
9         return this.x;
10    }
11    public U Second() {
12        return this.y;
13    }
14 }
15 Pair<int, boolean> p = new Pair<int, boolean>(10, true);
```

Declarations:	PC	this	x	y
	5	Pair	T	U

Classes:	Pair
	First=Pair → T
	Pair=(Pair × T × U) → Pair
	Second=Pair → U
	x=T
	y=U

Class generators: generics

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

```
1 class Pair<T, U> {
2     private T x;
3     private U y;
4     public Pair(T x, U y) {
5         this.x = x;
6         this.y = y;
7     }
8     public T First() {
9         return this.x;
10    }
11    public U Second() {
12        return this.y;
13    }
14 }
15 Pair<int, boolean> p = new Pair<int, boolean>(10, true);
```

Declarations:	PC	this	x	y
	6	Pair	T	U

Classes:	Pair
	First=Pair → T
	Pair=(Pair × T × U) → Pair
	Second=Pair → U
	x=T
	y=U

Class generators: generics

Generics

The INFDEV team

Introduction

Arrays as a simple generic data type

Class generators: generics

Interfaces and generics

Generic lists: a concrete example

Lambda functions

Conclusion

```
1 class Pair<T, U> {
2     private T x;
3     private U y;
4     public Pair(T x, U y) {
5         this.x = x;
6         this.y = y;
7     }
8     public T First() {
9         return this.x;
10    }
11    public U Second() {
12        return this.y;
13    }
14 }
15 Pair<int, boolean> p = new Pair<int, boolean>(10, true);
```

Declarations:	PC	this	x	y
	7	Pair	T	U

Classes:	Pair
	First=Pair → T
	Pair=(Pair × T × U) → Pair
	Second=Pair → U
	x=T
	y=U

Class generators: generics

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

```
1 class Pair<T, U> {
2     private T x;
3     private U y;
4     public Pair(T x, U y) {
5         this.x = x;
6         this.y = y;
7     }
8     public T First() {
9         return this.x;
10    }
11    public U Second() {
12        return this.y;
13    }
14 }
15 Pair<int, boolean> p = new Pair<int, boolean>(10, true);
```

Declarations:

PC	this
10	Pair

Classes:

Pair
First=Pair → T
Pair=(Pair × T × U) → Pair
Second=Pair → U
x=T
y=U

Class generators: generics

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

```
1 class Pair<T, U> {
2     private T x;
3     private U y;
4     public Pair(T x, U y) {
5         this.x = x;
6         this.y = y;
7     }
8     public T First() {
9         return this.x;
10    }
11    public U Second() {
12        return this.y;
13    }
14 }
15 Pair<int, boolean> p = new Pair<int, boolean>(10, true);
```

Declarations:	PC	this
	12	Pair

Classes:	Pair
	First=Pair → T
	Pair=(Pair × T × U) → Pair
	Second=Pair → U
	x=T
	y=U

Class generators: generics

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

```
1 class Pair<T, U> {
2     private T x;
3     private U y;
4     public Pair(T x, U y) {
5         this.x = x;
6         this.y = y;
7     }
8     public T First() {
9         return this.x;
10    }
11    public U Second() {
12        return this.y;
13    }
14 }
15 Pair<int, boolean> p = new Pair<int, boolean>(10, true);
```

Declarations:

PC
15

Classes:

Pair
First=Pair → T
Pair=(Pair × T × U) → Pair
Second=Pair → U
x=T
y=U

Class generators: generics

Generics

The INFDEV team

Introduction

Arrays as a simple generic data type

Class generators: generics

Interfaces and generics

Generic lists: a concrete example

Lambda functions

Conclusion

```
1 class Pair<T, U> {
2     private T x;
3     private U y;
4     public Pair(T x, U y) {
5         this.x = x;
6         this.y = y;
7     }
8     public T First() {
9         return this.x;
10    }
11    public U Second() {
12        return this.y;
13    }
14 }
15 Pair<int, bool> p = new Pair<int, bool>(10, true);
```

Declarations:	PC	P
	16	Pair<int, bool>

Classes:	Pair
	First=Pair → T
	Pair=(Pair × T × U) → Pair
	Second=Pair → U
	x=T
	y=U

Class generators: generics

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

This means that the compiler would actually generate code that behaves like the following:

```
1 class PairIntBool {  
2     private x int;  
3     private y bool;  
4     public PairIntBool(int x,boolean y) {  
5         this.x = x;  
6         this.y = y;  
7     }  
8     public int First() {  
9         return this.x;  
10    }  
11    public bool Second() {  
12        return this.y;  
13    }  
14}  
15 PairIntBool p = new PairIntBool(10,true);
```

Class generators: generics

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

Which in Java then becomes:

```
1 class PairIntBool {  
2     private x int;  
3     private y bool;  
4     public PairIntBool(int x,boolean y) {  
5         this.x = x;  
6         this.y = y;  
7     }  
8     public int First() {  
9         return this.x;  
10    }  
11    public bool Second() {  
12        return this.y;  
13    }  
14 }  
15 PairIntBool p = new PairIntBool(10,true);
```

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

Interfaces and generics

Interfaces and generics

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

Introduction

- Interfaces can also be defined generically.
- When implementing a generic interface, we need to provide the types of its generic arguments.

Interfaces and generics

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

We can define a generic interface as follows:

```
1 interface IPair<T, U> {
2     T First();
3     U Second();
4 }
```

Interfaces and generics

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

Which in Java then becomes:

```
1 interface IPair<T, U> {  
2     T First();  
3     U Second();  
4 }
```

Interfaces and generics

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

The generic interface can then be implemented by a class
(generic or not) as follows:

```
1 ...  
2 class Pair<T, U> : IPair<T,U> {  
3     private x T;  
4     private y U;  
5     public Pair(T x,U y) {  
6         this.x = x;  
7         this.y = y;  
8     }  
9     public T First() {  
10        return this.x;  
11    }  
12    public U Second() {  
13        return this.y;  
14    }  
15 }  
16 IPair<int, bool> p = new Pair<int, bool>(10,true);
```

Interfaces and generics

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

Which in Java then becomes:

```
1 ...
2 class Pair<T, U> implements IPair<T,U> {
3     private x T;
4     private y U;
5     public Pair(T x,U y) {
6         this.x = x;
7         this.y = y;
8     }
9     public T First() {
10        return this.x;
11    }
12    public U Second() {
13        return this.y;
14    }
15 }
16 IPair<Integer, Boolean> p = new Pair<Integer, Boolean>(10,true);
```

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

Generic lists: a concrete example

Generic lists: a concrete example

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

Ingredients

- We need an `List<T>` generic interface
- We then need two generic classes that implement the interface: `Empty<T>` and `Node<T>`
- In `Empty` the methods fail with a (descriptive) error.

Generic lists: a concrete example

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

Ingredients

Live coding demo: generic lists.

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

Lambda functions

Lambda functions

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

Introduction

- Both C# and (since very recently) Java feature anonymous functions
- They are very handy whenever we need to implement an interface with a single method, such as `PerformAction` or similar.

Lambda functions

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

In C# lambda functions have types:

- `Func<T, U>` for a function that takes as input a parameter of type T, and which returns a value of type U
 - `Func<T1, T2, U>` for a function that takes as input parameters of type T1 and T2, and which returns a value of type U
 - ...
-
- `Action<T>` for a function that takes as input a parameter of type T, and which returns nothing (`void`)
 - `Action<T1, T2>` for a function that takes as input parameters of type T1 and T2, and which returns nothing (`void`)
 - ...

Lambda functions

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

In Java lambda functions have many more types (see the documentation for the full list):

- Function<T,R> for a function that takes as input a parameter of type T, and which returns a value of type R
- BiFunction<T1,T2,R> for a function that takes as input parameters of type T1 and T2, and which returns a value of type R
- Predicate<T> for a function that takes as input a parameter of type T, and which returns a boolean value
- ...

Lambda functions

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

Declaration of lambda functions is quite simple: the parameters are separated from the body by an ASCII arrow.

Calling lambda functions is also simple: just brackets with the argument in C#, and an appropriate method in Java (see documentation).

```
1 Func<int,int> f = x => (x + 2);  
2 Console.WriteLine(f(10));
```

Lambda functions

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

```
1 Func<int,int> f = x => (x + 2);  
2 Console.WriteLine(f(10));
```

Stack:

PC
1

Lambda functions

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

```
1 Func<int,int> f = x => (x + 2);  
2 Console.WriteLine(f(10));
```

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

Stack:

PC	f
2	(x) => return (x + 2);

Lambda functions

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

```
1 Func<int,int> f = x => (x + 2);  
2 Console.WriteLine(f(10));
```

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

Stack:

PC	...		ret	x
2	...		null	10

Lambda functions

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

```
1 Func<int,int> f = x => (x + 2);  
2 Console.WriteLine(f(10));
```

Stack:

PC	...	ret
2	...	12

Lambda functions

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

```
1 Func<int,int> f = x => (x + 2);  
2 Console.WriteLine(f(10));
```

Stack:

PC	f
3	(x) => return (x + 2);

Output:

12

Lambda functions

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

Introduction

Live coding demo: generic lists with map, filter, and fold.

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

Conclusion

Conclusion

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

Looking back

- Generics make it possible to define a class once, but use it with multiple types as arguments
- It is particularly useful for containers such as arrays, tuples, lists, etc.
- It is particularly useful for relationships such as functions

This is it!

Generics

The INFDEV
team

Introduction

Arrays as a
simple
generic data
type

Class
generators:
generics

Interfaces
and generics

Generic lists:
a concrete
example

Lambda
functions

Conclusion

The best of luck, and thanks for the
attention!