

תוכנה 1

תרגול 12: Wildcards וירחשה
נעמה מאיר ומתי שמרת

תזכורת

■ מערכים הם קו-וריאנטים

■ אם Sub הוא תת-טיפוס של Super אז Sub[] הוא תת-טיפוס של Super[]

✓
`Sub[] sub = ...
Super[] sup = sub;`

■ טיפוסים גנריים הם וריאנטים

■ אם T1 ו T2 טיפוסים שונים אז, לדוגמה, בין הטיפוסים List<T1> ו List<T2> לא מתקיים יחס של תתי-טיפוסים גם אם יחס כזה מתקיים בין T1 ו T2

✗
`List<Sub> sub = new ArrayList<sub>();
List<Super> sup = sub;`

מחסנית

נתונה המחלקה: ■

```
public class Stack<E> {  
    public Stack() {...}  
    public void push(E e) {...}  
    public E pop() {...}  
    public boolean isEmpty() {...}  
}
```

נרצה להוסיף ■

```
public void pushAll(Collection<E> src) {  
    for (E e : src)  
        push(e);  
}
```

מה הבעיה במימוש? ■

הבעיה

■ מה קורה עבור הקוד הבא:

■ זיכרו Integer יורש מ Number

```
Stack<Number> numberStack = new Stack<Number>();  
Collection<Integer> integers = ...  
numberStack.pushAll(integers);
```

■ הודעת שגיאה

The method pushAll(Collection<Number>) in the type Stack<Number> is not applicable for the arguments (Collection<Integer>)

■ ממה נובעת הודעת השגיאה?

פתרון - Wildcards

■ שלושה סוגים של wildcards:

1. ?

קבוצת "כל הטיפוסים" או "טיפוס כלשהו"

2. **T extends ?**

משפחת תתי הטיפוס של T (כולל T)

3. **T super ?**

משפחת טיפוס העל של T (כולל T)

? extends E

טיפוס הקלט ל `pushAll` ■

במקום "Collection of E" נרצה ■

"Collection of **some subtype of E**"

```
public class Stack<E> {  
    ...  
    public void pushAll(Collection<? extends E> src) {  
        for (E e : src)  
            push(e);  
    }  
}
```

חסם עליון על טיפוס הקלט ■

E הוא תת טיפוס של עצמו ■

popAll

כעת נרצה להוסיף את popAll ■

```
public class Stack<E> {  
    ...  
    public void popAll(Collection<E> dst) {  
        while (!isEmpty())  
            dst.add(pop());  
    }  
}
```

בעיית קומפילציה? ■

מה עם קוד הלקוח? ■

קוד הלקוח

האם יש בעיה בקוד הלקוח? ■

✓ `Stack<Number> numberStack = new Stack<Number>();`
`Object o = numberStack.pop();`

✗ `Collection<Object> objects = ...`
`numberStack.popAll(objects);`

האם השימוש ב `extend` מתאים גם פה? ■

? super E

טיפוס הקלט ל popAll ■

במקום "Collection of E" נרצה ■

"Collection of **some supertype of E**"

```
public class Stack<E> {  
    ...  
    public void popAll(Collection<? super E> dst) {  
        while (!isEmpty())  
            dst.add(pop());  
    }  
}
```

חסם תחתון על טיפוס הקלט ■

E הוא תת טיפוס של עצמו ■

get-put principal*

■ השתמשו ב **extends** כאשר אתם קוראים נתונים ממבנה, ב **super** כאשר אתם מכניסים נתונים למבנה ואל תשתמשו ב wildcards כאשר אתם עושים את שניהם

■ ב pushAll קוראים נתונים מהמשתנה src

■ ב popAll מכניסים נתונים למשתנה dst

* “*Java Generics and Collections*” by Naftalin and Wadler

Method Overloading & Overriding

```
public class A {  
    public float foo(float a, float b) throws IOException{  
    }  
}
```

```
public class B extends A {  
    ...  
}
```

Which of the following methods can be defined in B:

1. `float foo(float a, float b){...}`
2. `public int foo(int a, int b) throws Exception{...}`
3. `public float foo(float a, float b) throws Exception{...}`
4. `public float foo(float p, float q) {...}`

Answer: 2 and 4

Method Overriding

```
public class A {  
    public void print() {  
        System.out.println("A");  
    }  
}
```

```
public class B extends A {  
    public void print(){  
        System.out.println("B");  
    }  
}
```

```
public class C {  
    public static void main(...) {  
        B b = new B();  
        A a = b;  
  
        b.print();  
        a.print();  
    }  
}
```

The output is:	compile? If not, why?
B	throw an exception?
B	no, what is the output?

Method Overriding & Visibility

```
public class A {  
    public void print() {  
        System.out.println("A");  
    }  
}
```

```
public class B extends A {  
    protected void print() {  
        System.out.println("B");  
    }  
}
```

```
public class C {  
    public static void main(String[] args) {  
        B b = new B();  
        b.print();  
    }  
}
```

Compilation error:
"Cannot reduce the visibility
of the inherited method"

no, why?
time exception?
the output?

Method Overriding & Visibility

```
public class A {  
    protected void print() {  
        System.out.println("A");  
    }  
}
```

```
public class B extends A {  
    public void print() {  
        System.out.println("B");  
    }  
}
```

```
public class C {  
    public static void main(String[] args) {  
        B b = new B();  
        b.print();  
    }  
}
```

The output is:

B

?

Inheritance

```
public class A {  
  
    public void foo() {  
        System.out.println("A.foo()");  
    }  
  
    public void bar() {  
        System.out.println("A.bar()");  
        foo();  
    }  
}
```

```
public class B extends A {  
  
    public void foo() {  
        System.out.println("B.foo()");  
    }  
  
    public static void main(...) {  
        A a = new B();  
        a.bar();  
    }  
}
```

The output is:

A.bar()

B.foo()

file? If not, why?

a runtime exception?

what is the output?

Inheritance

```
public class A {  
    private void foo() {  
        System.out.println("A.foo()");  
    }  
  
    public void bar() {  
        System.out.println("A.bar()");  
        foo();  
    }  
}
```

```
public class B extends A {  
    public void foo() {  
        System.out.println("B.foo()");  
    }  
  
    public static void main(String[] args) {  
        A a = new B();  
        a.bar();  
    }  
}
```

Does the code compile? If no, why?
Does the code throw a runtime exception?
If yes, why? If no, what is the output?