

Java Generics

Parametric Polymorphism

SUPER-WILDCARD GENERICS

DR. ERIC CHOU

IEEE SENIOR MEMBER



Super Wildcard

- When is the wildcard **<? super T>** needed? Consider the example in **SuperWildcardDemo.java**. The example creates a stack of strings in **stack1** (line 3) and a stack of objects in **stack2** (line 4), and invokes **add(stack1, stack2)** (line 8) to add the strings in **stack1** into **stack2**.
- **GenericStack<? super T>** is used to declare **stack2** in line 13. If **<? super T>** is replaced by **<T>**, a compile error will occur on **add(stack1, stack2)** in line 8, because **stack1**'s type is **GenericStack<String>** and **stack2**'s type is **GenericStack<Object>**. **<? super T>** represents type **T** or a supertype of **T**. Object is a supertype of **String**.



Wildcard Hierarchy

This program will also work if the method header in lines 12–13 is modified as follows:

```
public static <T> void add(GenericStack<? extends T> stack1, GenericStack<T> stack2)
```

The inheritance relationship involving generic types and wildcard types is summarized in Figure D. In this figure, **A** and **B** represent classes or interfaces, and **E** is a generic type parameter.

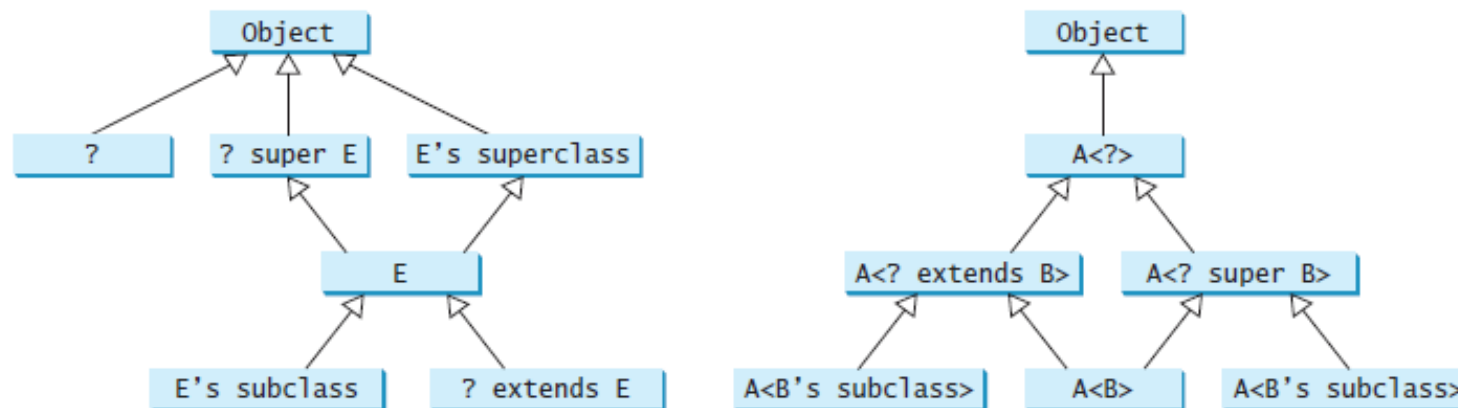


Figure D.



Super Wildcard

Demo Program: SuperWildCardDemo.java

Go BlueJ!