Generic Programming

Generic programming

- Defining constructs that can be used with different data types
- I.e., using same code for different data types

Implemented in Java through

1. Inheritance → A extends B
2. Type variables → <A>
Generic Programming Examples

Inheritance

```java
Class A {
    doWork( A x ) { … } 
}
Class B extends A { … }

A w1 = new A( );
B w2 = new B( );
w1.doWork( w1 );
w2.doWork( w2 );
```

Type Variables

```java
Class W<T> {
    doWork( T x ) { … }
}
Class A { … } 
Class B { … }

W<A> x1 = new W<A>( );
W<B> x2 = new W<B>( );
A w1 = new A( );
B w2 = new B( );
x1.doWork( w1 );
x2.doWork( w2 );
```

doWork( ) applied to objects of both class A and B
**Generic Class**

- Class with one or more type variables
  - Example → class `ArrayList<E>`

- To use generic class, provide an actual type
  - **Valid types**
    - Class → `ArrayList<String>`
    - Interface → `ArrayList<Comparable>`
  - **Invalid types**
    - Primitive type → `ArrayList<int>` (use wrappers) → `ArrayList<Integer>`
Defining a Generic Class

- Append type variable(s) to class name
  - Use angle brackets → ClassName<type variable>
- Can use any name for type variable
  - But typically single uppercase letter → E, K, V, etc…
- Use the type variable to define
  - Type of variables
  - Type of method parameters
  - Method return type
  - Object allocation
- Arrays
  - Type of an array object may not be a type variable or a parameterized type, unless it is an unbounded wildcard type
  - How to define arrays?
    - T[] data = (T[]) new Object[size];
Example Generic Class

Example

```java
public class myGeneric<T> {
    private T value;
    public myGeneric( T v ) { value = v; }
    public T getVal( ) { return value; }
    public void setVal( T newV ) { value = newV; }
}
```

Example (Queue class in queue package)
Generics and Subtyping

In general if B is a subtype of A, and GT is a generic type declaration it is not the case that GT<B> is a subtype of GT<A>

Example

```java
ArrayList<String> strL = new ArrayList<String>();
ArrayList<Object> objL = strL; // Illegal!
```
Wildcards

? (unknown)

- Collection<?>
  - Collection whose element type matches anything

Bounded Wildcard

- Example: ArrayList<? extends Shape>
  - Unknown type that is Shape or subtype of Shape

Summary

- <? \(\rightarrow\) unknown type
- <? extends typeExpression\(\rightarrow\) unknown type that is typeExpression or a subtype of typeExpression
- <? super typeExpression\(\rightarrow\) unknown type that is typeExpression or a supertype of typeExpression.
- typeExpression can involve further occurrences of wildcard type expressions

Example (WildCard class in wildcard package)