제너릭 제한

제너릭 제한

❖ 타입 파라미터에 지정되는 구체적인 타입 제한할 필요

- 상속 및 구현 관계 이용해 타입 제한

```
public <T extends 상위타입> 리턴타입 메소드(매개변수, ...) { ... }
```

  - 상위 타입은 클래스 뿐만 아니라 인터페이스도 가능

- 타입 파라미터를 대체할 구체적인 타입
  - 상위타입이거나 하위 또는 구현 클래스만 지정 가능

```
public <T extends Number> int compare(T t1, T t2) {
  double v1 = t1.doubleValue();   //Number의 doubleValue() 메소드 사용
  double v2 = t2.doubleValue();   //Number의 doubleValue() 메소드 사용
  return Double.compare(v1, v2);
}
```

제너릭 제한

❖ Util.java

```
public class Util {
    public static <T extends Number> int compare(T t1, T t2) {
        double v1 = t1.doubleValue();
        // System.out.println(t1.getClass().getName());
        double v2 = t2.doubleValue();
        // System.out.println(t2.getClass().getName());
        return Double.compare(v1, v2);
    }
}
```

제너릭 제한

❖ BoundedTypeParameterExample.java

```
public class BoundedTypeParameterExample {
    public static void main(String[] args) {
        // String str = Util.compare("a", "b"); (x)

        int result1 = Util.compare(10, 20);
        System.out.println(result1);

        int result2 = Util.compare(4.5, 3);
        System.out.println(result2);
    }
}
```

제너릭 제한

❖ 와일드카드 타입의 세가지 형태

- 제네릭타입<?> : Unbounded Wildcards (제한없음)
  타입 파라미터를 대치하는 구체적인 타입으로 모든 클래스나 인터페이스 타입이 올 수 있다.

- 제네릭타입<? extends 상위타입> : Upper Bounded Wildcards (상위 클래스 제한)
  타입 파라미터를 대치하는 구체적인 타입으로 상위 타입이나 하위 타입만 올 수 있다.

- 제네릭타입<? super 하위타입> : Lower Bounded Wildcards (하위 클래스 제한)
  타입 파라미터를 대치하는 구체적인 타입으로 하위 타입이나 상위 타입이 올 수 있다.

제너릭 제한

❖ Course.java

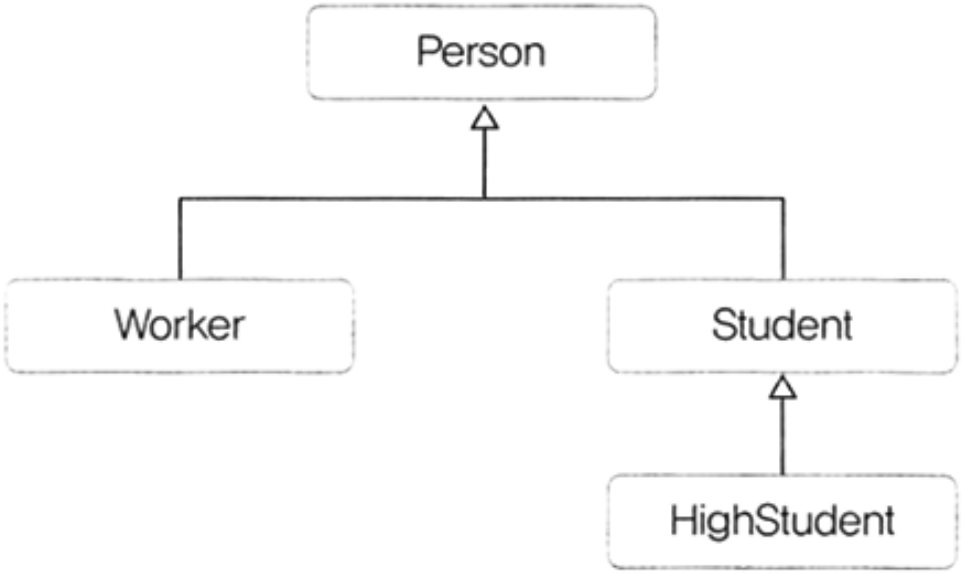
```
public class Course<T> {
    private String name;
    private T[] students;

    public Course(String name, int capacity) {
        this.name = name;
        students = (T[]) (new Object[capacity]);
    }

    public String getName() {
        return name;
    }

    public T[] getStudents() {
        return students;
    }
```

제너릭 제한

❖ Course.java

```
    public void add(T t) {
        for (int i = 0; i < students.length; i++) {
            if (students[i] == null) {
                students[i] = t;
                break;
            }
        }
    }
}
```

제너릭 제한



- Course<?>

  수강생은 모든 타입(Person, Worker, Student, HighStudent, Dog)이 될 수 있다.

- Course<? extends Student>

  수강생은 Student와 HighStudent만 될 수 있다.

- Course<? super Worker>

  수강생은 Worker와 Person만 될 수 있다.

제너릭 제한

❖ Person.java

```
public class Person {
    private String name;

    public Person(String name) {
        this.name = name;
    }

    public String getName() { return name; }
    public String toString() { return name; }
}
```

제너릭 제한

❖ Student.java

```
public class Student extends Person {
    public Student(String name) {
        super(name);
    }
}
```

❖ HighStudent.java

```
public class HighStudent extends Student {
    public HighStudent(String name) {
        super(name);
    }
}
```

❖ Worker.java

```
public class Worker extends Person {
    public Worker(String name) {
        super(name);
    }
}
```

제너릭 제한

❖ WildCardExample.java

```
import java.util.Arrays;

public class WildCardExample {
    public static void registerCourse(Course<?> course) {
        System.out.println(course.getName() + " 수강생: " +
                           Arrays.toString(course.getStudents()));
    }

    public static void registerCourseStudent(Course<? extends Student> course) {
        System.out.println(course.getName() + " 수강생: " +
                           Arrays.toString(course.getStudents()));
    }

    public static void registerCourseWorker(Course<? super Worker> course) {
        System.out.println(course.getName() + " 수강생: " +
                           Arrays.toString(course.getStudents()));
    }
}
```

제너릭 제한

❖ WildCardExample.java

```
        // registerCourseStudent(personCourse); (x)
        // registerCourseStudent(workerCourse); (x)
        registerCourseStudent(studentCourse);
        registerCourseStudent(highStudentCourse);
        System.out.println();

        registerCourseWorker(personCourse);
        registerCourseWorker(workerCourse);
        // registerCourseWorker(studentCourse); (x)
        // registerCourseWorker(highStudentCourse); (x)
    }
}
```