# Effective Java
# Item 34

haocheng

# Emulate extensible enums with interfaces

# Extensibility of enum types

```java
// Emulated extensible enum using an interface
public interface Operation {

    double apply(double x, double y);

}
```

```java
public enum BasicOperation implements Operation {
        PLUS("+") {
                public double apply(double x, double y) {
                        return x + y;
                }
        },
        MINUS("-") {
                public double apply(double x, double y) {
                        return x - y;
                }
        },
        TIMES("*") {
                public double apply(double x, double y) {
                        return x * y;
                }
        },
        DIVIDE("/") {
                public double apply(double x, double y) {
                        return x / y;
                }
        };

        private final String symbol;

        BasicOperation(String symbol) {
                this.symbol = symbol;
        }
}
```

```java
public enum ExtendedOperation implements Operation {
        EXP("^") {
                public double apply(double x, double y) {
                        return Math.pow(x, y);
                }
        },
        REMAINDER("%") {
                public double apply(double x, double y) {
                        return x % y;
                }
        };
        private final String symbol;

        ExtendedOperation(String symbol) {
                this.symbol = symbol;
        }

        @Override
        public String toString() {
                return symbol;
        }
}
```

# Bounded Type Token

```
1  public class OperationTest {
2
3          @Test
4          public void run_test1() throws Exception {
5                  double x = Double.parseDouble("2");
6                  double y = Double.parseDouble("4");
7                  test1(ExtendedOperation.class, x, y);
8          }
9
10         private <T extends Enum<T> & Operation> void test1(Class<T> opSet,
11                         double x, double y) {
12                 for (Operation op : opSet.getEnumConstants())
13                         System.out.printf("%f %s %f = %f%n", x, op, y, op.apply(x, y));
14         }
15
16 }
```

possible to pass in an entire extension enum type and use its elements

<T extends Enum<T> & Operation>

# Bounded Wildcard Type

```
1  public class OperationTest2 {
2
3          @Test
4          public void run_test2() throws Exception {
5                  double x = Double.parseDouble("2");
6                  double y = Double.parseDouble("4");
7                  test2(Arrays.asList(ExtendedOperation.values()), x, y);
8          }
9
10         private static void test2(Collection<? extends Operation> opSet, double x,
11                         double y) {
12                 for (Operation op : opSet)
13                         System.out.printf("%f %s %f = %f%n", x, op, y, op.apply(x, y));
14         }
15
16 }
```

## Pros

- Less Complex
- Combine operations from multiple implementation

## Cons

- Cannot use EnumSet/EnumMap

Cannot inherit implementations

Thank you!