Generics

```
public class Money<T> {
    private T currency;
    public T createT() {
        //Type parameter 'T' cannot be instantiated directly
        // return new T();
    }
}
```

Class and Methods with generics

```
class Ball<X>{
  //this does not compile because X cannot be referenced from a sta
 // public static <T> void catchBall(T t, X x) {}  DOES NOT COMPILE

  //this is how we can fix it, basically we re-define X
  public static <T, X> void catchBall(T t, X x) {}

  //or make the method instance, not static:
  public <T> void catchBall2(T t, X x) {}

  //here we redefine X so the type on Ball is hidden.
  public <X extends Number> void inflateBall(X t, X x) {}

}
```

Class and methods with generics

Diamond Operator

```
List<String> list = new ArrayList<>();
```

The diamond operator <> cannot be used on the left side of the assignment.

```
HashSet<> set = new HashSet<Object>();  //does not compile!
```

Methods with generics

Static Methods

```
    public static <T extends Number> void calculate(T t){}
```

WRONG: static needs to be defined before generics declaration!

```
    public  <T extends Number> static void calculate(T t){} //does NO
```

```
  private static <U extends Exception> void printMessage(U u) {
      System.out.println(u.getMessage());
  }

  public static void invokePrintMessage() {
      printMessage(new FileNotFoundException("not found"));
      //optional syntax for invoking a generic method
      StaticMethodGenerics.<NullPointerException>printMessage(new Nul
      ("D"));
  }
```

Static Methods Generics

Instance Methods

```
   public <T> T getFirstElement(T[] array) {/*...*/}
```

GenericsInClassAndMethods

Instantiate with generics

Example

Bounding Generic Types

Upper-Bounded wildcards

```
List<? extends Number> listOfInteger = new ArrayList<>();
//does not compile...what if it's a list of AtomicInteger
//listOfInteger.add(new Integer(3));

//does not compile...what if it's a list of Integer
//listOfInteger.add(new AtomicInteger(2));
```

We cannot add an element to a list declared s upper bounded wildcards!

Lower-Bounded wildcards

```
List<? super IOException> exceptions = new ArrayList<Exception>();
exceptions.add(new IOException("error"));// yes we can add it
exceptions.add(new FileNotFoundException("error"));// yes we can ad

//we cannot add an Exception as it could be a list of IOException a
//exceptions.add(new Exception("error"));
```

On the right side, the element should be a type whose super-class is defined as super at left side.

```
//List<? super IOException> ioExceptions = new ArrayList<FileNotFou
List<? super IOException> ioExceptions = new ArrayList<Exception>()
```

wildcards mutable and immutable collections

- wildcardList is immutable
- listExtends is immutable
- listSuper is mutable

```
List<?> wildcardList = new ArrayList<String>();
List<? extends IOException> listExtends = new ArrayList<FileNotF
List<? super IOException> listSuper = new ArrayList<Object>();
```

Example wildcard

Bounded with Classes (not collections)

```
class Cage<T extends Pet> {}
class Pet {}
class Dog extends Pet{}


//Cage<Pet> cagePets = new Cage<Dog>();  //DOES NOT Compile

  //upper-bounded cases
Cage<? extends Pet> upperBounded = new Cage<Pet>();
Cage<? extends Pet> upperBounded2 = new Cage<Dog>();

//lower-bounded
Cage<? super Pet> lowerBounded2 = new Cage<Pet>();
```

Classes With Lower and Upper Bounded

Inheritance of methods with generics

Generic Methods