```
template<typename T>
void swap(T n1, T n2){
T temp = n1;
n1 = n2;
n2 = temp;
}
```

Generics

Till java 1.4

```
class Box{
private Object obj;

public void setObj(Object obj){
        this.obj=obj;
}

public Object getObj(){
        return obj;
}
}
Box b1 = new Box();
b1.setObj(new Double(10.20));
Integer i1 =(Integer) b1.getObj();
```

From Java 1.5 onwards

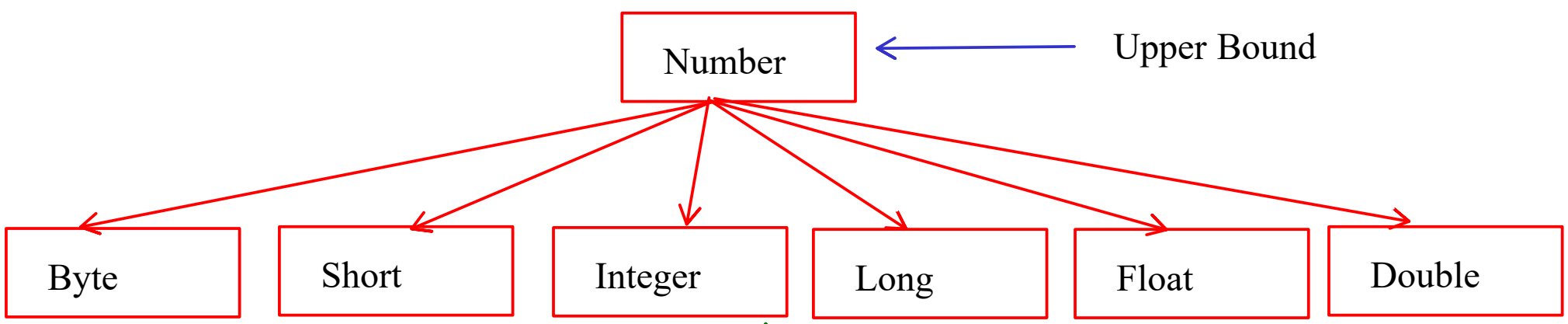
```
class Box<Type>{
private Type obj;

public void setObj(Type obj){
        this.obj=obj;
}

public Type getObj(){
        return obj;
}
}

Box<Integer> b1 = new Box<Integer>();
b1.setObj(new String(10.12));
Integer i1 = b1.getObj();
```

Type safety

Generics
Generic classes
Generic Methods
Generic Interfaces

Type Parameters

1. Bounded
   - It is used for classes
2. Unbounded
   - It is used for class references
   - wild card -> ?
   - We can set bounds on the Unbounded types
     1. Upper Bound
     2. Lower Bound



Box<? extends Number> b; // upper Bound

Box<? super Integer> b; // lower bound

```
class LinkedList<T>{
T data;
}
```

```
// Bounded type
class Box<T extends Number>

Box<? super Integer> b = new Box<Double>();
```

```
class Box{
private Object obj;

public void setObj(Object obj){
        this.obj=obj;
}

public Object getObj(){
        return obj;
}

}
```

Comparable
Comparator

```
Box b1 = new Box();
```

```
interface Comparable{
        int compareTo(Object o){
        }
}
```

```
interface Comparable<T>{
        int compareTo(T o){
        }
}
```

```
class Employee{

}

Employee e1= new Employee(1,"Anil",20000);
Employee e2 = new Employee(2,"Mukesh",20000);;
```

```
class Employee implements Comparable{
int compareTo(Object o){
Employee e =  (Employee) o;
int diff = this.salary - e.salary;
return diff;
}
}
```

```
class Employee implements Comparable<Employee>{
int compareTo(Employee o){
}
}

class Student implements Comparable<Student>{
int compareTo(Student o){
}
}

class Car implements Comparable<Car>{
int compareTo(Car o){
}
}
```

```
void sort(Object [] arr){
Comparable c = arr[0];

}
```

```
compare(Employee o1, Employee o2){
o1.name-o2.name;

}
```

```
interface Comparator<T>
{
      int compare(T o1, T o2);
}
```

given objects
o1, o2

```
class EmpComp implements Comparator{

}

Employee e1 = new Employee();
Employee e2 = new Employee();
Comparator c = new EmpComp();
c.compare(e1,e2);
```

```
<T>void method1(T ref){

}
```

```
interface Comparable<T>
{
      int compareTo(T o);
}
```

this -> given object

```
Employee e1 = new Employee();
Employee e2 = new Employee();
e1.compareTo(e2);
```

```
class Box<T>{
private T obj;
}

      public void unBoxing(Box<? extends Mobile> b){

      }

      main(){
      Box<Apple> b1 = new Box<>();
      Box<Samsung> b2 = new Box<>();
      Box<Lux> b3 = new Box<>();
      }
```