GENERIC PROGRAMMING

Generic Programming: creation of programming constructs that can be used with many different types.
ex.:

- ArrayList<String>
- ArrayList<Student>
- Instantiating a generic class ArrayList is a generic class

TYPE variable names
E - Element type in a collection
K - Key type in Map
V - Value Type in Map
T – General Type
S, U – Additional general type

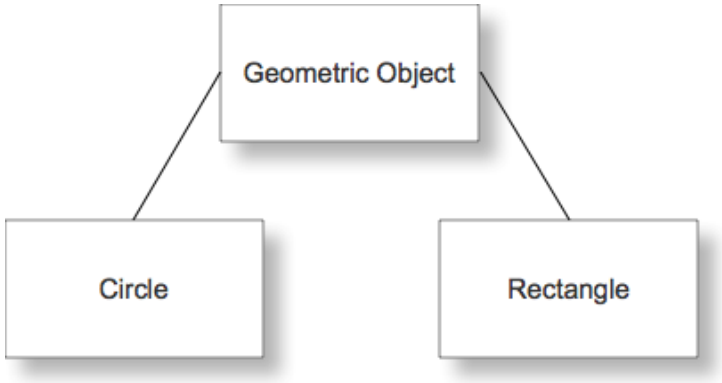
Define a generic class or interface and generic methods
GernericStack.java

```
public class GenericStack<E> {
        ArrayList<E> list = new ArrayList<E>();
        public int getsize() {
                return list.size();
        }
        public E peek() {
                return list.get(getsize() – 1);
        }
        Public void  push(E o) {
                list.add(o);
        }
        public E pop() {
                E o = list.get(getSize() – 1);
                list.remove(getSize() – 1);
                return o;
        }
```

```
        public boolean isEmpty()
        { return list.isEmpty();
        }
} //end class
```

Bounded generic type

A generic type can be specified as a subtype of another type




```
public Test {
        _____ main()
        {
                Circle c = new Circle(_____);
                Rectangle r = new Rectangle(____);
                System.out.println(Test.<GeometricObject>equalArea(c,r));
        }//end main
        public static <E extends GeometricObject> boolean equalArea(E o1, E o2)  {
                return o1.getArea() == o2.getArea();
        }
}
```

Wildcard Types

Name	Syntax	Meaning
Wildcard with lower bound	? extends T	Any subtype of T
Wildcard with upper bound	? super T	Any supertype of T
Unbounded wildcard	?	Anytype

Example: Using ? super T

```
public class Test {
        ____ main() {
                GenericStack<String> st1 = new GenericStack<String>();
                GenericStack<Object> st2 = new GenericStack<Object>();
                st1.push("a");
                st1.push("b");
                st2.push(2);
                add(st1, st2);
                …
        } //end main

public static <T> void add(GenericStack<T> st1, GenericStack<? super T> st2) {

                -------

                --------
}
```

Example: Using ? extends T

```
 main() {
        GenericStack<Integer> inStack = new GenericStack<Integer>();
        inStack.push(1);
        inStack.push(2);
        instack.push(-2);
        System.out.println(max(inStack));
}
public static double max(GenericStack<? extends Number> stack) //Integer is a subtype of
                                                                //Number
{
        …
        return max;
}
```

Example: Using ?

? is equivalent to <? extends Object>

_____ main() {

```
        GenericStack<Integer> st = new GenericStack<Integer>();
        st.push(1);
        st.push(5);
        st.push(3);
        print(st);
}//end main
public static void print(GenericStack<?> s) {
        Display all elements in the stack

}
```