Generics

Genericidade

- Generics introduces an extra level of abstraction in programming.
- The same code can be used for different data types.
- Again, generics requires that the programmer finds what is common in an algorithm and what changes according to the different data types that it can be used with.

Design

```
public class Point {
    private int x;
    private int y;
    …
}
```

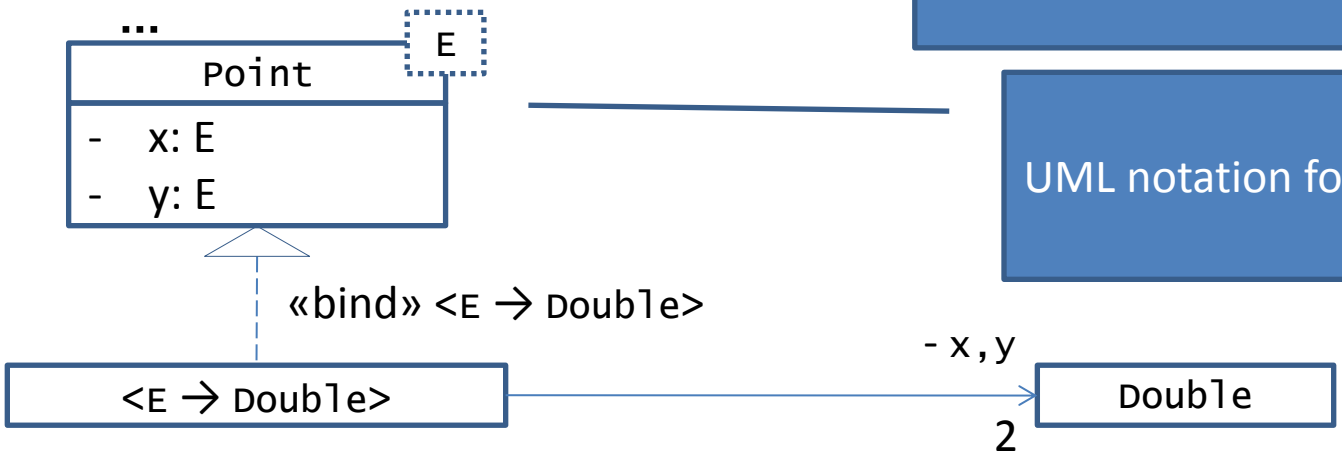
```
public class Point {
    private double x;
    private double y;
    …
}
```

Which is best?

Generic point

```
public class Point<E> {
    private E x;
    private E y;
    ...
}
```

Generic class. E is a parameter. Must be instantiated with a (non-primitive) type.



UML notation for generics.

Using a generic point

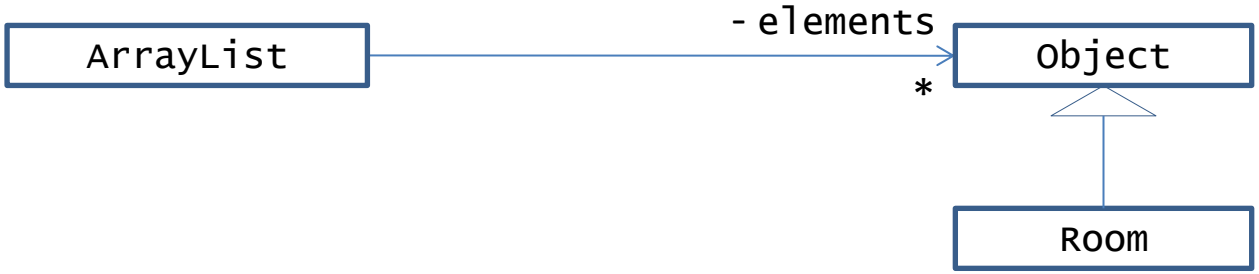
```
Point<Integer> a = new Point<Integer>(1, 2);

Point<Double> b = new Point<Double>(2.0, 1.0);
```

Point: implementing

```
public class Point<E> {

    private E x;
    private E y;

    public Point(E x, E y) {

      this.x = x;
      this.y = y;


    }
    …

}
```

Generic list?

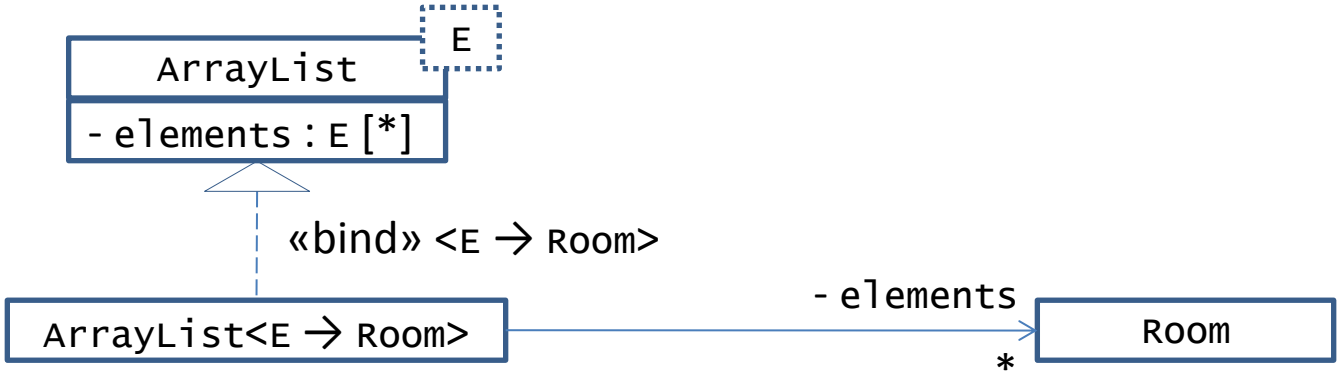
```
public class ArrayList {
    private Object[] elements;

    …
}
```



Generic list

```
public class ArrayList<E> {
    private E[] elements;

    …
}
```

Generics. This implementation is not possible in Java, although it is equivalent to the actual one.



Generic list

```
public class Floor {

    private ArrayList<Room> rooms =
        new ArrayList<Room>();

    public Floor(final int numberOfRooms) {
        for (int roomNumber = 1;
             roomNumber != numberOfRooms + 1;
             roomNumber++)
            rooms.addLast(new Room(roomNumber));
    }

    public void show() {
        while (rooms.hasNext())
            out.println(rooms.next());
    }

}
```

Generic list

```
public class ArrayList<E> {

    private E[] elements;
    private numberOfElements;

    public void addLast(E element) {
       if (full()) {
          duplicateSize();
       }
       elements[numberOfElements] = element;
       numberOfElements++;
    }

    …

}
```

References

- Y. Daniel Liang, *Introduction to Java Programming*, 7.ª edição, Prentice-Hall, 2008.

Summary

- Generics