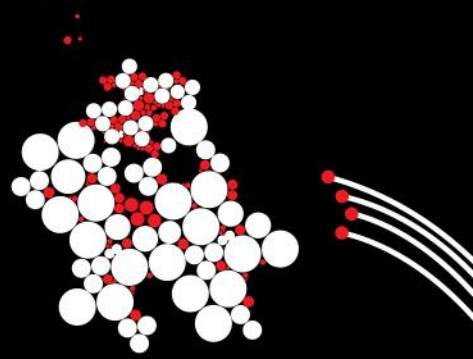
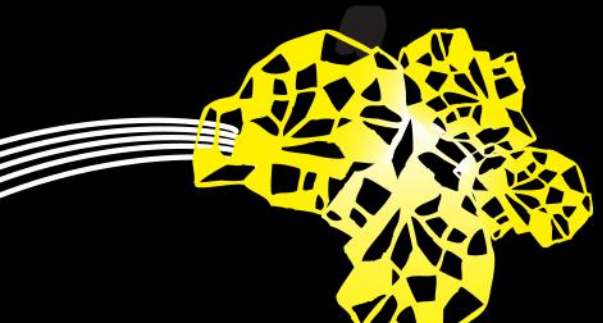
UNIVERSITY OF TWENTE.



Generics

Topic of Software Systems (TCS module 2)

Lecturer: Marieke Huisman



GENERICS

MOTIVATION

- Earlier video: list of Students with methods
- We want to reuse the list methods to make lists of elements of different types (Room, Date, etc.)
- Generics allow an interface or class to have an element type as a ‘parameter’, to be replaced by an ‘argument type’ when the object of the class is created
- By defining the List<E> interface (E is a ‘type parameter’) we can use the same list implementations for lists of objects of different types
  - List<Student>, List<Room>, List<Date>, etc.

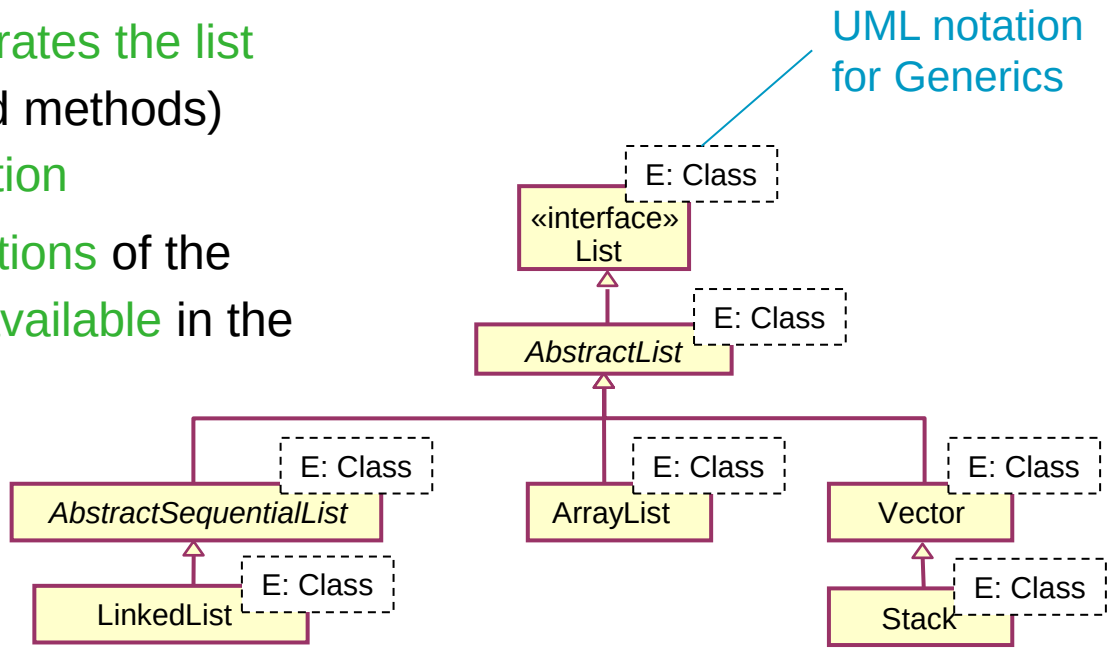
LIST OF PRIMITIVE VALUES

- Generics classes and interfaces(e.g., List<E> interface) only support reference types, not primitive types like int, float, boolean, char, etc.
- Java offers wrapper (reference) types for primitive types

  int → Integer
  double → Double
  char → Character
- List of integer values is denoted as List<Integer>
- List of double values is denoted as List<Double>, etc.

List<E> IMPLEMENTATIONS

- List<E> interface separates the list concept (and related methods) from its implementation
- Various implementations of the List<E> interface are available in the java.utils package



USE OF GENERICS

- Java Collection classes
- Any class that you would like to define over a generic type

- Instantiating a generic type

```
List<Student> sl1 = new ArrayList<Student>();

class MyOption<E> { ..}

MyOption<Object> x = new MyOption<Object>();
```