Dasar Dasar Pemrograman 2

Acuan: Introduction to Java Programming anda Data Structure, Bab 19 dan Oracle Java Tutorial on Generics

Sumber Slide: Liang,

Dimodifikasi untuk Fasilkom UI oleh Ade Azurat

Topik 13: Generics

Why Do You Get a Warning?

```
public class ShowUncheckedWarning {
  public static void main(String[] args) {
    java.util.ArrayList list =
         new java.util.ArrayList();
    list.add("Java Programming");
  }
}
```

ShowUncheckedWarning

Type safety: The method add(Object) belongs to the raw type ArrayList. References to generic type ArrayList<E> should be parameterized

To understand the compile warning on this line, you need to learn JDK 1.6 generics.

FACULTY OF COMPUTER SCIENCE

UNIVERSITAS INDONESIA

Veritas, Probitas, Iustitia

Fix the Warning

```
public class ShowUncheckedWarning {
  public static void main(String[] args) {
    java.util.ArrayList<String> list =
      new java.util.ArrayList<String>();
    list.add("Java Programming");
  }
}
```

No compile warning on this line.

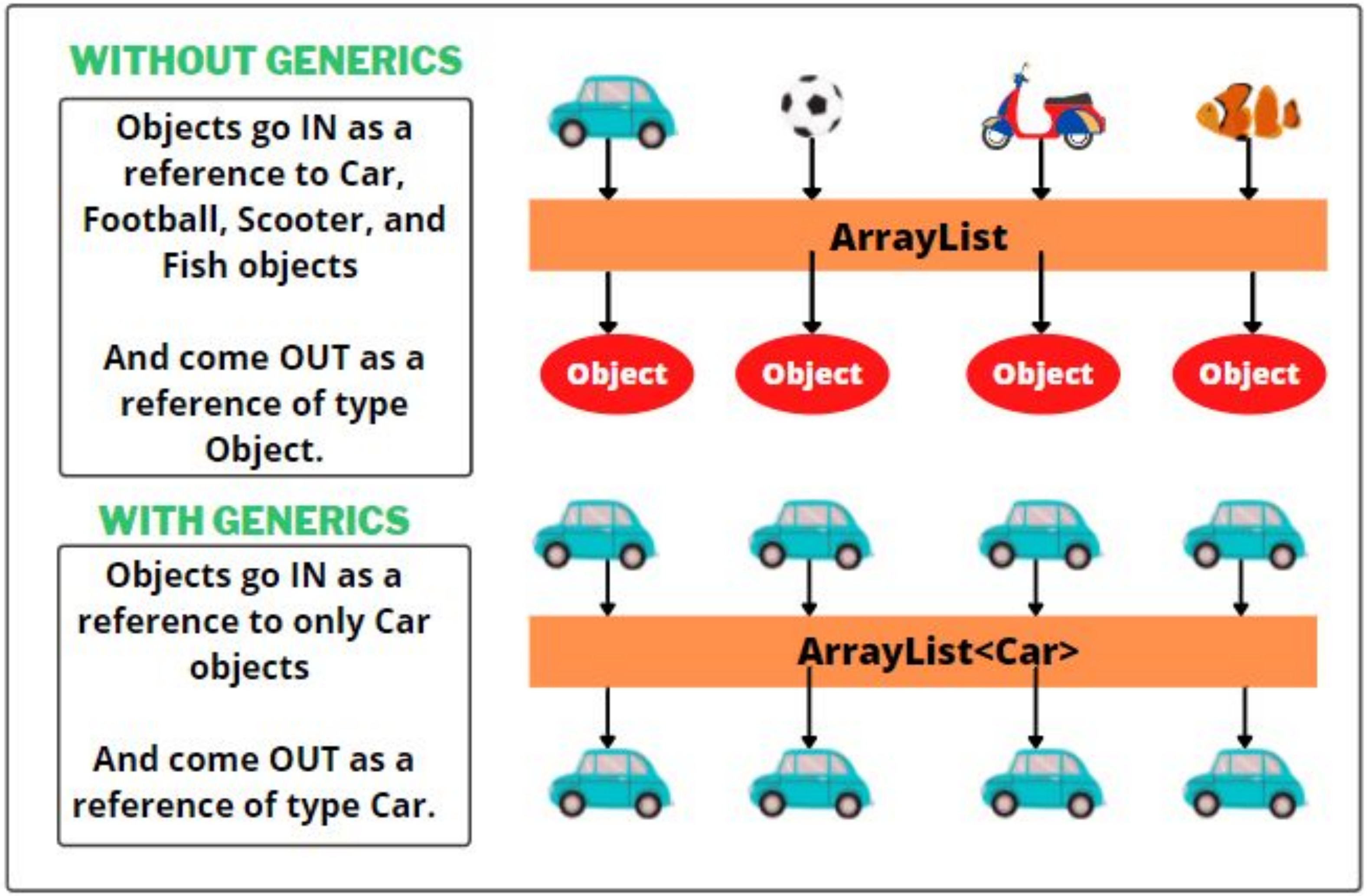
Compile ok , run-time error!

```
public class FruitDemo{
        public static void main(String[] argv){
                Fruit fruit1 = new Apple();
                Fruit fruit2 = new Orange();
                Fruit fruit3 = new Fruit();
                Apple apple;
                apple = (Apple) fruit1;
                apple = (Apple) fruit2; // Compile ok, runtime error!
                apple = (Apple) fruit3; // Compile ok, runtime error!
        }
}
```

Penyelesaian dengan menggunakan `instanceof` tidak *applicable* untuk skala besar!
Kita tidak ingin memunculkan runtime error dulu baru kemudian menambahkan `instanceof`. Kita ingin bisa mengetahui ini saat compile time.

Motivation

https://www.javacodegeeks.com/2015/03/polymorphism-in-java-generics.html
https://www.scientecheasy.com/2021/10/generics-in-java.html


WITHOUT GENERICS
Objects go IN as a reference to Car, Football, Scooter, and Fish objects
And come OUT as a reference of type Object.
ArrayList
Object
Object
Object
Object
WITH GENERICS
Objects go IN as a reference to only Car objects
And come OUT as a reference of type Car.
ArrayList<Car>

Objectives

- ❑ To know the benefits of generics (§19.1).
- ❑ To use generic classes and interfaces (§19.2).
- ❑ To declare generic classes and interfaces (§19.3).
- ❑ To understand why generic types can improve reliability and readability (§19.3).
- ❑ To declare and use generic methods and bounded generic types (§19.4).
- ❑ To use raw types for backward compatibility (§19.5).
- ❑ To know wildcard types and understand why they are necessary (§19.6).
- ❑ To convert legacy code using JDK 1.5 generics (§19.7).
- ❑ To understand that generic type information is erased by the compiler and all instances of a generic class share the same runtime class file (§19.8).
- ❑ To know certain restrictions on generic types caused by type erasure (§19.8).
- ❑ To design and implement generic matrix classes (§19.9).

What is Generics?

- *Generics* is the capability to parameterize types.
- With this capability, you can define a class or a method with generic types that can be substituted using concrete types by the compiler.
- For example you may define a generic stack class that stores the elements of a generic type.
- From this generic class, you may create a stack object for holding strings and a stack object for holding numbers. Here, strings and numbers are concrete types that replace the generic type.

Why Generics?

- ✔ The key benefit of generics is to enable errors to be detected at **compile time** rather than at **runtime**.
  - ❑ A generic class or method permits you to specify allowable types of objects that the class or method may work with.
  - ❑ If you attempt to use the class or method with an **incompatible object**, a **compile error** occurs.

Generic Type

```
package java.lang;

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

(a) Prior to JDK 1.5

```
package java.lang;

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

(b) JDK 1.5

Generic Instantiation

Runtime error

```
Comparable c = new Date();
System.out.println(c.compareTo("red"));
```

(a) Prior to JDK 1.5

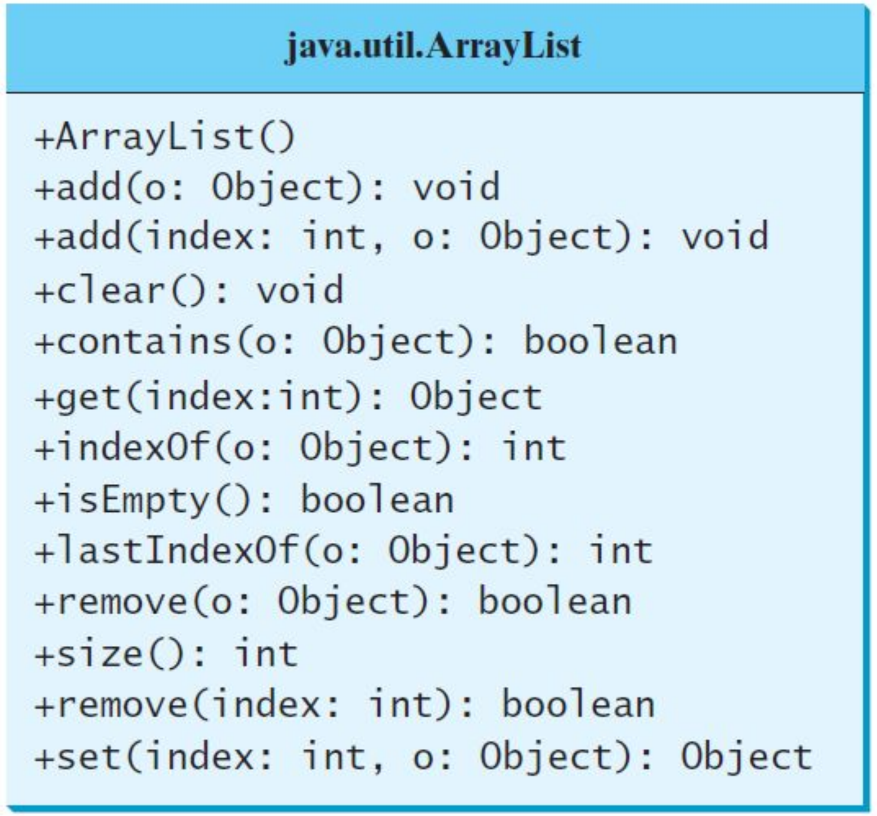
```
Comparable<Date> c = new Date();
System.out.println(c.compareTo("red"));
```

(b) JDK 1.5

Improves reliability

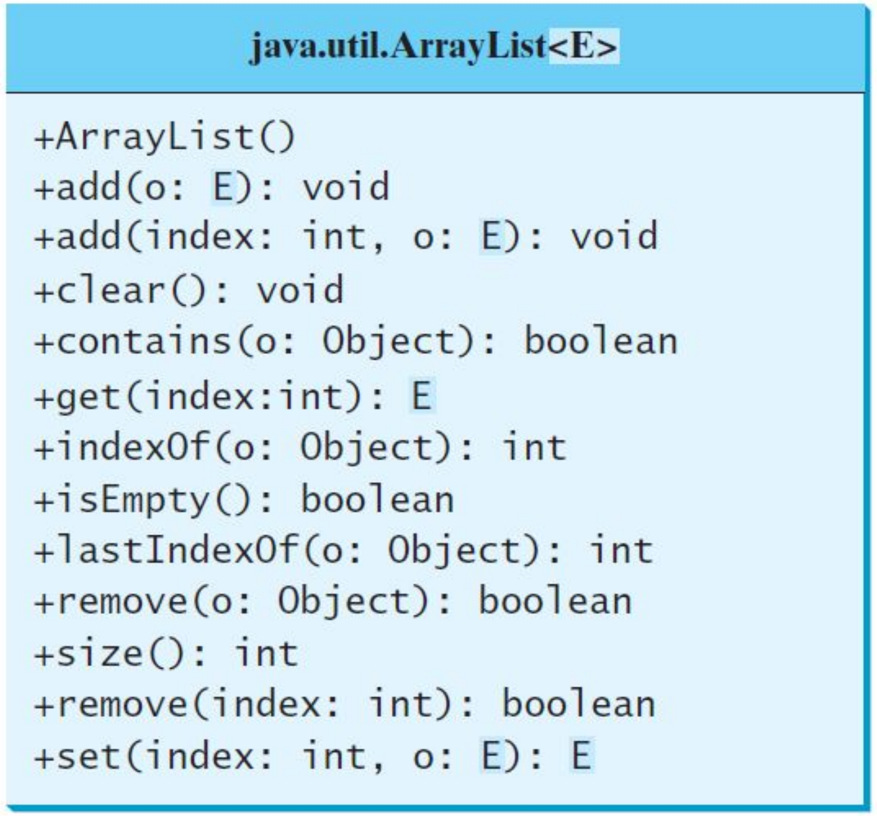
Compile error

Generic ArrayList in JDK 1.5



```
java.util.ArrayList
-------------------------------------------
+ArrayList()
+add(o: Object): void
+add(index: int, o: Object): void
+clear(): void
+contains(o: Object): boolean
+get(index:int): Object
+indexOf(o: Object): int
+isEmpty(): boolean
+lastIndexOf(o: Object): int
+remove(o: Object): boolean
+size(): int
+remove(index: int): boolean
+set(index: int, o: Object): Object
```

(a) ArrayList before JDK 1.5



```
java.util.ArrayList<E>
-------------------------------------------
+ArrayList()
+add(o: E): void
+add(index: int, o: E): void
+clear(): void
+contains(o: Object): boolean
+get(index:int): E
+indexOf(o: Object): int
+isEmpty(): boolean
+lastIndexOf(o: Object): int
+remove(o: Object): boolean
+size(): int
+remove(index: int): boolean
+set(index: int, o: E): E
```

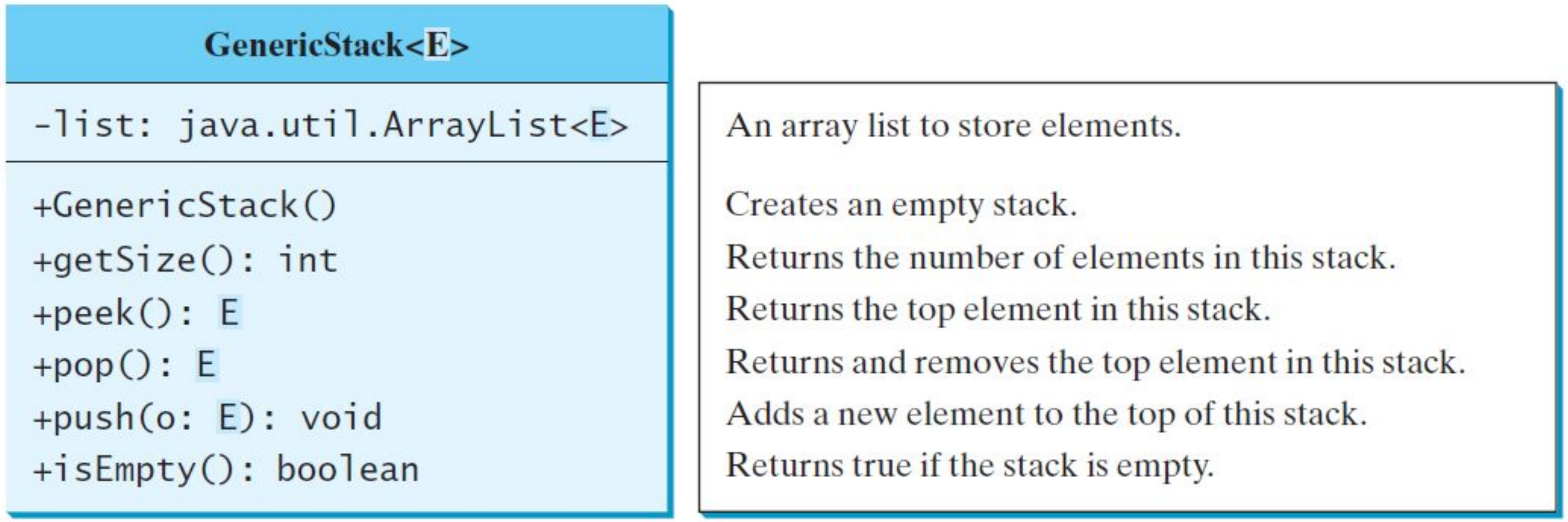
(b) ArrayList since JDK 1.5

No Casting Needed

```
ArrayList<Double> list = new ArrayList<>();
list.add(5.5); // 5.5 is automatically converted to new Double(5.5)
list.add(3.0); // 3.0 is automatically converted to new Double(3.0)
Double doubleObject = list.get(0); // No casting is needed
double d = list.get(1); // Automatically converted to double
```

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia
FACULTY OF COMPUTER SCIENCE

Declaring Generic Classes and Interfaces



GenericStack<E>	
-list: java.util.ArrayList<E>	An array list to store elements.
+GenericStack()	Creates an empty stack.
+getSize(): int	Returns the number of elements in this stack.
+peek(): E	Returns the top element in this stack.
+pop(): E	Returns and removes the top element in this stack.
+push(o: E): void	Adds a new element to the top of this stack.
+isEmpty(): boolean	Returns true if the stack is empty.

GenericStack

FACULTY OF COMPUTER SCIENCE

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia

Raw Type and Backward Compatibility

```
// raw type
ArrayList list = new ArrayList();


// This is roughly equivalent to

ArrayList<Object> list = new ArrayList<Object>();
```

FACULTY OF COMPUTER SCIENCE

Raw Type is Unsafe

```
// Max.java: Find a maximum object
public class Max {
  /** Return the maximum between two objects */
  public static Comparable max(Comparable o1, Comparable o2) {
    if (o1.compareTo(o2) > 0)
      return o1;
    else
      return o2;
  }
}
```

max("Welcome", 23);   **Runtime Error!**

Make it Safe

```
// MaxUsingGenericType.java: Find a maximum object
public class MaxUsingGenericType {
  /** Return the maximum between two objects */
  public static <E extends Comparable<E>> E max(E o1, E o2) {
    if (o1.compareTo(o2) > 0)
      return o1;
    else
      return o2;
  }
}
```

Compile Time Error!

max("Welcome", 23);

Lebih baik! Karena Programmer bisa tahu dan memperbaiki lebih awal!

Avoiding Unsafe Raw Types

Use

```
new ArrayList<ConcreteType>()
```

Instead of

```
new ArrayList();
```

TestArrayListNew

FACULTY OF COMPUTER SCIENCE

UNIVERSITAS INDONESIA

Veritas, Probitas, Iustitia

Generic Methods

```
public static void print(Object[] list) {
    for (int i = 0; i < list.length; i++)
      System.out.print(list[i] + " ");
    System.out.println();
  }

public static <E> void print(E[] list) {
    for (int i = 0; i < list.length; i++)
      System.out.print(list[i] + " ");
    System.out.println();
  }
```

Bounded Generic Type

```
public static void main(String[] args ) {
    Rectangle rectangle = new Rectangle(2, 2);
    Circle circle = new Circle (2);
    System.out.println("Same area? " + equalArea(rectangle, circle));
}

public static boolean equalArea(GeometricObject object1, GeometricObject object2) {
    return object1.getArea() == object2.getArea();
}


public static <E extends GeometricObject> boolean equalArea(E object1, E object2) {
    return object1.getArea() == object2.getArea();
}
```

GeometricObjectDemo.java

Wildcards

Why wildcards are necessary? See this example.

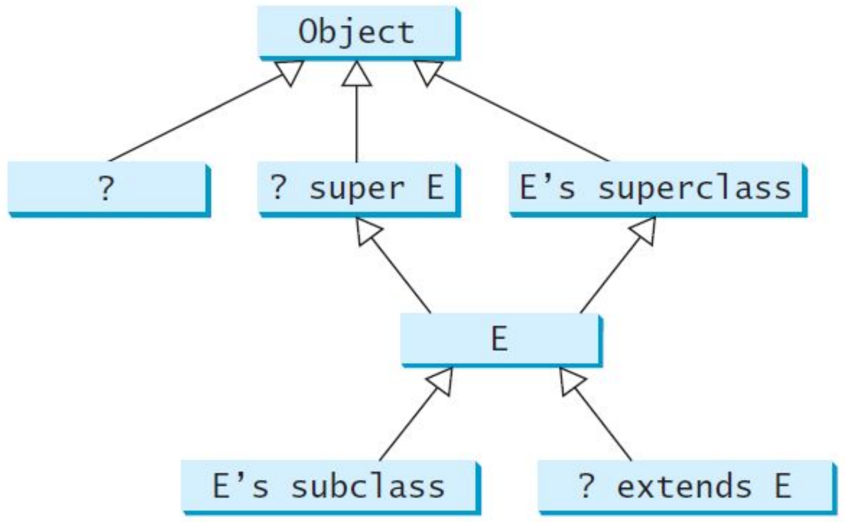
WildCardNeedDemo

?	unbounded wildcard
? extends T	bounded wildcard
? super T	lower bound wildcard

AnyWildCardDemo

SuperWildCardDemo

Generic Types and Wildcard Types



Erasure and Restrictions on Generics

- Generics are implemented using an approach called *type **erasure***.
- The compiler uses the generic type information to compile the code, but **erases** it afterwards.
- So the generic information is **not available** at run time.
- This approach enables the generic code to be backward-compatible with the legacy code that uses raw types.

Compile Time Checking

For example, the compiler checks whether generics is used correctly for the following code in (a) and translates it into the equivalent code in (b) for runtime use. The code in (b) uses the raw type.

```
ArrayList<String> list = new ArrayList<>();
list.add("Oklahoma");
String state = list.get(0);
```

(a)

```
ArrayList list = new ArrayList();
list.add("Oklahoma");
String state = (String)(list.get(0));
```

(b)

Important Facts

It is important to note that a generic class is shared by all its instances regardless of its actual generic type.

```
GenericStack<String> stack1 = new GenericStack<>();
GenericStack<Integer> stack2 = new GenericStack<>();
```

Although GenericStack<String> and GenericStack<Integer> are two types, but there is only one class GenericStack loaded into the JVM.

Restrictions on Generics

- ❑ Restriction 1: Cannot Create an Instance of a Generic Type. (i.e., new E()).
- ❑ Restriction 2: Generic Array Creation is Not Allowed. (i.e., new E[100]).
- ❑ Restriction 3: A Generic Type Parameter of a Class Is Not Allowed in a Static Context.
- ❑ Restriction 4: Exception Classes Cannot be Generic.

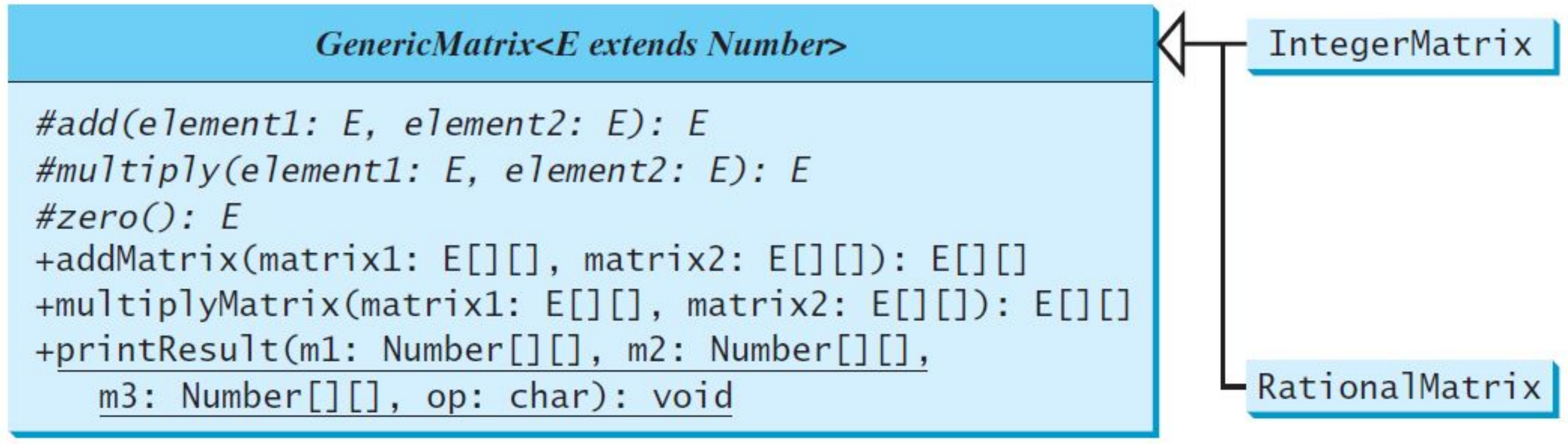
Designing Generic Matrix Classes

Objective: This example gives a generic class for matrix arithmetic. This class implements matrix addition and multiplication common for all types of matrices.

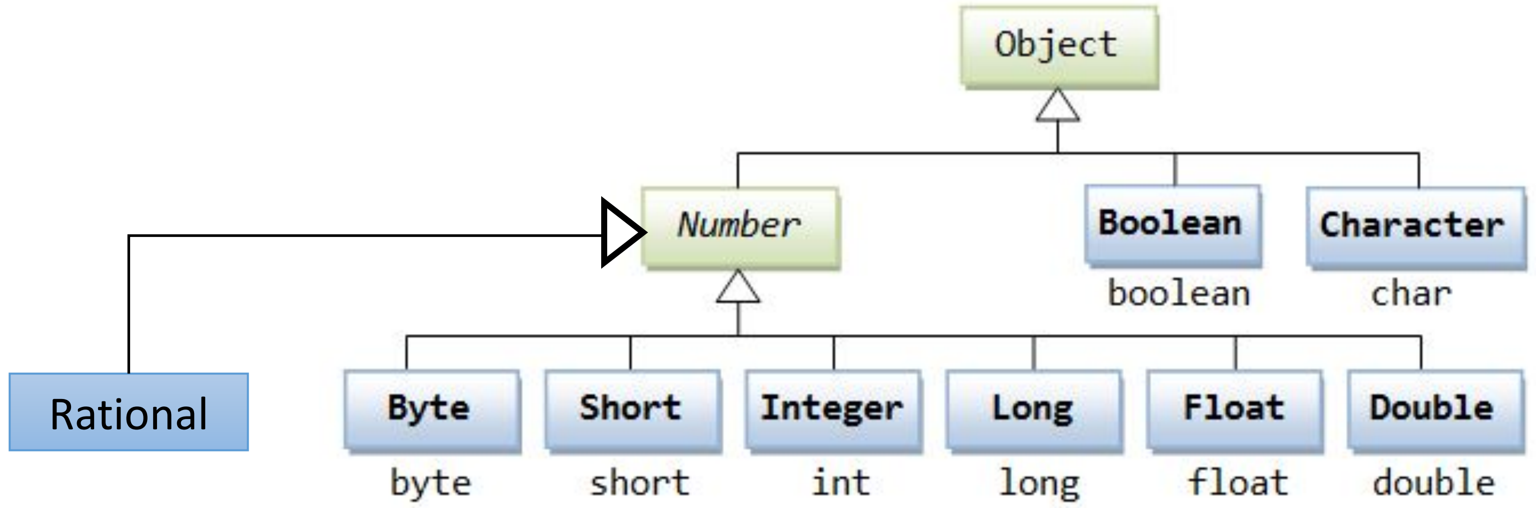
GenericMatrix

UML Class Diagram

GenericMatrix



GenericMatrix<E extends Number>
#add(element1: E, element2: E): E
#multiply(element1: E, element2: E): E
#zero(): E
+addMatrix(matrix1: E[][], matrix2: E[][]): E[][]
+multiplyMatrix(matrix1: E[][], matrix2: E[][]): E[][]
+printResult(m1: Number[][], m2: Number[][], m3: Number[][], op: char): void
IntegerMatrix
RationalMatrix



Generic Matrices

Objective: This example gives two programs that utilize the GenericMatrix class for integer matrix arithmetic and rational matrix arithmetic.

IntegerMatrix TestIntegerMatrix

RationalMatrix TestRationalMatrix

Ada Pertanyaan?

- ❑ Jadi apa sih generics itu?
- ❑ Buat apa generics, kalau tanpa generics juga bisa?
- ❑ Memang kalau error nya saat runtime kenapa?
- ❑ erasure? type nya dihapus?! hmm, jadi buat apa dong kalau nanti dihapus juga?

Selamat Berlatih!

Perhatikan lagi List Objective yang perlu dikuasai pekan ini.

Mari mencoba dan mempelajari nya di repl

Bisa fork beberapa contoh dari: https://repl.it/@AdeAzurat/Pekan11-Generics

Baca buku acuan dan berlatih!
Bila masih belum yakin tanyakan ke dosen, tutor atau Kak Burhan.

Semangat !

Persiapan Ujian Akhir Semester (20-12-2024)

- Perhatikan kembali, kerjakan kembali secara mandiri TP 3 dan TP 4!
- Pastikan anda memahami, bisa membuat kembali dan bisa meng-extends.
- Pahami konsep, inheritance, abstraction, encapsulation, polymorphism.
- Pahami konsep, abstract class dan inheritance, serta generic.
- Pahami pula konsep OO Design Principles dan SOLID.
- Materi ujian akhir mencakup materi dari awal kuliah dengan penekanan pada bagian setelah UTS.
- Selamat mempersiapkan diri untuk ujian.

Latihan: rebuild GenericMatrics

- Buat kelas GenericMatrics yang baru from scratch.
- Buat kelas Rational.java
- Jalankan dengan berkas TestRationalMatrix.java