Due to the print book page limit, we cannot inlcude all good CheckPoint questions in the physical book. The CheckPoint on this Website may contain extra questions not printed in the book. The questions in some sections may have been reordered as a result. Nevertheless, it is easy to find the CheckPoint questions in the book on this Website. Please send suggestions and errata to Dr. Liang at y.daniel.liang@gmail.com. Indicate the book, edition, and question number in your email. Thanks!

# Chapter 19 Check Point Questions

## Section 19.2

### ▼19.2.1

Are there any compile errors in (a) and (b)?

```
(a) Prior to JDK 1.5
ArrayList dates = new ArrayList();
dates.add(new Date());
dates.add(new String());

(b) Since JDK 1.5
ArrayList<Date> dates = new ArrayList<>();
dates.add(new Date());
dates.add(new String());
```

(a) will compile fine, but (b) has a compilation error on Line 3, because dates is declared as a list of Date objects. You cannot assign a string to the list.

Hide Answer

### ▼19.2.2

What is wrong in (a)? Is the code in (b) correct?

```
(a) Prior to JDK 1.5
ArrayList dates = new ArrayList();
dates.add(new Date());
Date date = dates.get(0);

(b) Since JDK 1.5
ArrayList<Date> dates = new ArrayList<>();
dates.add(new Date());
Date date = dates.get(0);
```

Casting is needed in (a), but no casting is necessary in (b) with the generic type ArrayList<Date> .

Hide Answer

### ▼19.2.3

What are the benefits of using generic types?

One important benefit is improving reliability and robustness. Potential errors can be detected by the compiler.

Hide Answer

## Section 19.3

### ▼19.3.1

What is the generic definition for java.lang.Comparable in the Java API?

```
package java.lang;

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

Hide Answer

**▼19.3.2**

Since you create an instance of ArrayList of strings using new ArrayList<String>(), should the constructor in the ArrayList class be defined as

```
public ArrayList<E>()
```

No.

Hide Answer

**▼19.3.3**

Can a generic class have multiple generic parameters?

Yes.

Hide Answer

**▼19.3.4**

How do you declare a generic type in a class?

To declare a generic type for a class, place the generic type after the class name, such as GenericStack<E>. To declare a generic type for a method, place the generic type for the method return type, such as <E> void max(E o1, E o2).

Hide Answer

## Section 19.4

**▼19.4.1**

How do you declare a generic method? How do you invoke a generic method?

To declare a generic method, you place the generic type <E> immediately after the keyword static in the method. A generic method can be invoked just like a regular method. The compiler automatically discovers the actual type.

Hide Answer

**▼19.4.2**

What is a bounded generic type?

Bounded generic type such as <E extends AClass> specifies that a generic type must be a subclass of AClass.

Hide Answer

## Section 19.5

**▼19.5.1**

Given int[] list = {1, 2, -1}, can you invoke sort(list) using the sort method in Listing 19.4?

No, because list is of type int[], but the sort method requires E[], where E is an object type.

Hide Answer

**▼19.5.2**

Given int[] list = {new Integer(1), new Integer(2), new Integer(-1)}, can you invoke sort(list) using the sort method in Listing 19.4?

No, because list is still of type int[], but the sort method requires E[], where E is an object type.

Hide Answer

## Section 19.6

**▼19.6.1**

What is a raw type? Why is a raw type unsafe? Why is the raw type allowed in Java?

When you use generic type without specifying an actual parameter, it is called a raw type. A raw type is unsafe, because some errors cannot be detected by the compiler. The raw type is allowed in Java for backward compatibility.

Hide Answer

**▼19.6.2**

What is the syntax to declare an ArrayList reference variable using the raw type and assign a raw type ArrayList object to it?

```
ArrayList list = new ArrayList();
```

Hide Answer

## Section 19.7

**▼19.7.1**

Is GenericStack the same as GenericStack<Object>?

GenericStack is roughly equivalent to GenericStack<Object>, but they are not the same. GenericStack<Object> is a generic instantiation, but GenericStack is a raw type.

Hide Answer

**▼19.7.2**

What are an unbounded wildcard, a bounded wildcard, and a lower-bound wildcard?

```
? is unbounded wildcard
? extends T is bounded wildcard
? super T is lower bounded wildcard
```

Hide Answer

**▼19.7.3**

What happens if lines 12-13 in Listing 19.9 are changed to

```
public static <T> void add(GenericStack<T> stack1,
  GenericStack<T> stack2)
```

The program cannot be compiled, because the element type in stack1 is GenericStack<String>, but

the element type is stack2 is GenericStack<Object>. add(stack1, stack2) cannot be matched.

Hide Answer

**▼19.7.4**

What happens if lines 12-13 in Listing 19.9 are changed to

```
public static <T> void add(GenericStack<? extends T> stack1,
  GenericStack<T> stack2)
```

The program can be compiled and run fine.

Hide Answer

## Section 19.8

**▼19.8.1**

What is erasure? Why are Java generics implemented using erasure?

Generic type information is used by the compiler to check whether the type is used safely. Afterwards the type information is erased. The type information is not available at runtime. This approach enables the generic code to be backward-compatible with the legacy code that uses raw types.

Hide Answer

**▼19.8.2**

If your program uses ArrayList<String> and ArrayList<Date> , does the JVM load both of them?

No. Only ArrayList is loaded.

Hide Answer

**▼19.8.3**

Can you create an instance using new E() for a generic type E? Why?

No, because the type information is not available at runtime.

Hide Answer

**▼19.8.4**

Can a method that uses a generic class parameter be static? Why?

Since all instances of a generic class have the same runtime class, the static variables and methods of a generic class is shared by all its instances. Therefore, it is illegal to refer a generic type parameter for a class in a static method or initializer.

Hide Answer

**▼19.8.5**

Can you define a custom generic exception class? Why?

No. The JVM have to check the exception thrown from the try clause to see if it matches the type specified in a catch clause. This is impossible, because the type information is not present at runtime.

Hide Answer

Section 19.9

**▼19.9.1**

Why are the add, multiple, and zero methods defined abstract in the GenericMatrix class?

Because these methods cannot be implemented in the GenericMatric class.

Hide Answer

**▼19.9.2**

How are the add, multiple, and zero methods implemented in the IntegerMatrix class?

In the IntegerMatrix class, the add method is implemented by adding the two numbers using the + operator. The multiply method is implemented by multiplying the two numbers using the * operator. The zero method is implemented to return 0.

Hide Answer

**▼19.9.3**

How are the add, multiple, and zero methods implemented in the RationalMatrix class?

In the RationalMatrix class, the add method is implemented by adding the two numbers using the add method in the Rational class. The multiply method is implemented by multiplying the two numbers using the multiply method in the Rational class. The zero method is implemented to return new Rational(0, 1).

Hide Answer

**▼19.9.4**

What would be wrong if the printResult method defined as follows?

```
public static void printResult(
  E[][] m1, E[][] m2, E[][] m3, char op)
```

You have to define it using:

```
public static <T> void printResult(
  T[][] m1, T[][] m2, T[][] m3, char op)
```

Hide Answer