

## Chapter 19 Generics

### Section 19.2 Motivations and Benefits

**19.1** Which of the following statements is correct?

- ☒ A. Generics can help detect type errors at compile time, thus make programs more robust.
- ☐ B. Generics can make programs easy to read.
- ☐ C. Generics can avoid cumbersome castings.
- ☐ D. Generics can make programs run faster.

Your answer A is incorrect  
The correct answer is ABC



**19.2** Fill in the code in Comparable\_\_\_\_\_ c = new Date();

- ☒ A. <String>
- ☐ B. <?>
- ☐ C. <Date>
- ☐ D. <E>

Your answer A is incorrect  
The correct answer is C



**19.3** Which of the following statements is correct?

- ☒ A. Comparable<String> c = new String("abc");
- ☐ B. Comparable<String> c = "abc";
- ☐ C. Comparable<String> c = new Date();
- ☐ D. Comparable<Object> c = new Date();

Your answer A is incorrect  
The correct answer is AB



**19.4** Suppose List list = new ArrayList(). Which of the following operations are correct?

- ☒ A. list.add("Red");
- ☐ B. list.add(new Integer(100));

- ☐ C. `list.add(new java.util.Date());`
- ☐ D. `list.add(new ArrayList());`

Your answer A is incorrect



The correct answer is ABCD

**19.5** Suppose `List<String> list = new ArrayList<String>`. Which of the following operations are correct?

- ☒ A. `list.add("Red");`
- ☐ B. `list.add(new Integer(100));`
- ☐ C. `list.add(new java.util.Date());`
- ☐ D. `list.add(new ArrayList());`

Your answer is correct



**19.6** Suppose `ArrayList<Double>list = new ArrayList<>()`. Which of the following statements are correct?

- ☒ A. `list.add(5.5);` // 5.5 is automatically converted to new `Double(5.5)`
- ☐ B. `list.add(3.0);` // 3.0 is automatically converted to new `Double(3.0)`
- ☐ C. `Double doubleObject = list.get(0);` // No casting is needed
- ☐ D. `double d = list.get(1);` // Automatically converted to double

Your answer A is incorrect



The correct answer is ABCD

### Section 19.3 Declaring Generic Classes and Interfaces

**19.7** To declare a class named A with a generic type, use

- ☒ A. `public class A<E> { ... }`
- ☐ B. `public class A<E, F> { ... }`
- ☐ C. `public class A(E) { ... }`
- ☐ D. `public class A(E, F) { ... }`


Your answer is correct



**19.8** To declare a class named A with two generic types, use


- ☒ A. `public class A<E> { ... }`

- ☐ B. `public class A<E, F> { ... }`
- ☐ C. `public class A(E) { ... }`
- ☐ D. `public class A(E, F) { ... }`

Your answer A is incorrect   
The correct answer is B


**19.9** To declare an interface named A with a generic type, use

- ☒ A. `public interface A<E> { ... }`
- ☐ B. `public interface A<E, F> { ... }`
- ☐ C. `public interface A(E) { ... }`
- ☐ D. `public interface A(E, F) { ... }`

Your answer is correct 


**19.10** To declare an interface named A with two generic types, use

- ☒ A. `public interface A<E> { ... }`
- ☐ B. `public interface A<E, F> { ... }`
- ☐ C. `public interface A(E) { ... }`
- ☐ D. `public interface A(E, F) { ... }`

Your answer A is incorrect   
The correct answer is B

**19.11** To create a list to store integers, use

- ☒ A. `ArrayList<Object> list = new ArrayList<>();`
- ☐ B. `ArrayList<Integer> list = new ArrayList<>();`
- ☐ C. `ArrayList<int> list = new ArrayList<int>();`
- ☐ D. `ArrayList<Number> list = new ArrayList<>();`

Your answer A is incorrect   
The correct answer is B

### Section 19.4 Generic Methods

**19.12** The method header is left blank in the following code. Fill in the header.

```


public class GenericMethodDemo {
    public static void main(String[] args ) {
        Integer[] integers = {1, 2, 3, 4, 5};
        String[] strings = {"London", "Paris", "New York", "Austin"};

        print(integers);
        print(strings);
    }

    _____ {
        for (int i = 0; i < list.length; i++)
            System.out.print(list[i] + " ");
        System.out.println();
    }
}

```

- ☒ A. public static void print(Integer[] list)
- ☐ B. public static void print(String[] list)
- ☐ C. public static void print(int[] list)
- ☐ D. public static void print(Object[] list)
- ☐ E. public static <E> void print(E[] list)

Your answer A is incorrect   
 The correct answer is DE

**19.13** To create a generic type bounded by Number, use

- ☒ A. <E extends Number>
- ☐ B. <E extends Object>
- ☐ C. <E>
- ☐ D. <E extends Integer>

Your answer is correct 

### Section 19.6 Raw Type and Backward Compatibility

**19.14** Which of the following declarations use raw type?

- ☒ A. ArrayList<Object> list = new ArrayList<>();
- ☐ B. ArrayList<String> list = new ArrayList<>();
- ☐ C. ArrayList<Integer> list = new ArrayList<>();
- ☐ D. ArrayList list = new ArrayList();

Your answer A is incorrect 

The correct answer is D

**19.15** If you use the javac command to compile a program that contains raw type, what would the compiler do?

- ☒ A. report syntax error
- ☐ B. report warning and generate a class file
- ☐ C. report warning without generating a class file
- ☐ D. no error and generate a class file
- ☐ E. report warning and generate a class file if no other errors in the program.

Your answer A is incorrect



The correct answer is E

Explanation: For javac, a class file is generated even if the program has compile warnings.

**19.16** If you use the javac ?Xlint:unchecked command to compile a program that contains raw type, what would the compiler do?

- ☒ A. report compile error
- ☐ B. report warning and generate a class file
- ☐ C. report warning without generating a class file
- ☐ D. no error and generate a class file

Your answer A is incorrect



The correct answer is B

### Section 19.7 Wildcards

**19.17** Is ArrayList<Integer> a subclass of ArrayList<Object>?

- ☒ A. Yes
- ☐ B. No

Your answer A is incorrect



The correct answer is B

**19.18** Is ArrayList<Integer> a subclass of ArrayList<?>?

- ☒ A. Yes
- ☐ B. No

Your answer is correct



**19.19** Is `ArrayList<Integer>` a subclass of `ArrayList<? extends Number>`?

- ☒ A. Yes
- ☐ B. No

Your answer is correct



**19.20** Is `ArrayList<Number>` a subclass of `ArrayList<? extends Number>`?

- ☒ A. Yes
- ☐ B. No

Your answer is correct



**19.21** Is `ArrayList<?>` same as `ArrayList<? extends Object>`?

- ☒ A. Yes
- ☐ B. No

Your answer is correct



**19.22** Does `<? super Number>` represent a superclass of `Number`?

- ☒ A. Yes
- ☐ B. No

Your answer is correct



**19.23** Which of the following can be used to replace `YYYYYYYY` in the following code?

```
public class WildCardDemo3 {
    public static void main(String[] args) {
        GenericStack<String> stack1 = new GenericStack<>();
        GenericStack<Object> stack2 = new GenericStack<>();
        stack2.push("Java");
        stack2.push(2);
        stack1.push("Sun");
        add(stack1, stack2);
        WildCardDemo2.print(stack2);
    }

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

```

        while (!stack1.isEmpty())
            stack2.push(stack1.pop());
    }
}

```

- ☒ A. ? super Object
- ☐ B. ? super T
- ☐ C. ? extends T
- ☐ D. ? extends Object

Your answer A is incorrect  
The correct answer is B



**19.24** Which of the following can be used to replace YYYYYYYY in the following code?

```

public class WildCardDemo3 {
    public static void main(String[] args) {
        GenericStack<String> stack1 = new GenericStack<>();
        GenericStack<Object> stack2 = new GenericStack<>();
        stack2.push("Java");
        stack2.push(2);
        stack1.push("Sun");
        add(stack1, stack2);
        WildCardDemo2.print(stack2);
    }

    public static <T> void YYYYYYYY {
        while (!stack1.isEmpty())
            stack2.push(stack1.pop());
    }
}

```

- ☒ A. add(GenericStack<T> stack1, GenericStack<T> stack2)
- ☐ B. add(GenericStack<? extends T> stack1, GenericStack<T> stack2)
- ☐ C. add(GenericStack<T> stack1, GenericStack<? super T> stack2)
- ☐ D. add(GenericStack<T> stack1, GenericStack<Object> stack2)

Your answer A is incorrect  
The correct answer is BC



### Section 19.8 Erasure and Restrictions on Generics

**19.25** ArrayList<String> and ArrayList<Integer> are two types. Does the JVM load two classes ArrayList<String> and ArrayList<Integer>?

- ☒ A. Yes

☐ B. No

Your answer A is incorrect



The correct answer is B

Explanation: The JVM loads just one ArrayList.

**19.26** Which of the following statements are true?

- ☒ A. Generic type information is present at compile time.
- ☐ B. Generic type information is not present at runtime.
- ☐ C. You cannot create an instance using a generic class type parameter.
- ☐ D. You cannot create an array using a generic class type parameter.
- ☐ E. You cannot create an array using a generic class.

Your answer A is incorrect



The correct answer is ABCDE

**19.27** If E is a generic type for a class, can E be referenced from a static method?

- ☒ A. Yes
- ☐ B. No

Your answer A is incorrect



The correct answer is B

Explanation: It is illegal to refer to a generic type parameter for a class in a static method or initializer, because generic type for a class belongs to a specific instantiation of the class.

**19.28** Fill in the most appropriate code in the blanks in the MyInt class?

```
public class MyInt implements _____ {
    int id;

    public MyInt(int id) {
        this.id = id;
    }

    public String toString() {
        return String.valueOf(id);
    }

    public int compareTo(_____ arg0) {
        if (id > arg0.id)
```



```
        return 1;
    else if (id < arg0.id)
        return -1;
    else
        return 0;
    }
}
```

- ☒ A. Comparable / Object
- ☐ B. Comparable<MyInt> / MyInt
- ☐ C. Comparable<MyInt> / Object
- ☐ D. Comparable / MyInt

Your answer A is incorrect  
The correct answer is B

