**Chapter 19 Check Point Questions**

1. **Are there any compile errors in the following code.**
2. Prior to JDK 1.5

ArrayList dates = new ArrayList();

dates.add(new Date());

dates.add(new String());

1. Yes B. **No**
2. **Are there any compile errors in the following code.**

ArrayList<Date> dates = new ArrayList<>();

dates.add(new Date());

dates.add(new String());

1. **Yes** B. No

Explanation: It 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.

1. **What is wrong in the following code?**

ArrayList dates = new ArrayList();

dates.add(new Date());

Date date = dates.get(0);

1. No wrong. B. **Casting is needed** C. No Casting is needed D. None of the Above.
2. **What is wrong in the following code?**

ArrayList<Date> dates = new ArrayList<>();

dates.add(new Date());

Date date = dates.get(0);

1. No wrong. B. Casting is needed C. **No Casting is needed** D. None of the Above.
2. **What are the benefits of using generic types?**
3. One important benefit is improving reliability and robustness.
4. Potential errors can be detected by the compiler.
5. Both A and B
6. None of the above.
7. **What is the generic definition for java.lang.Comparable in the Java API?**
8. **package java.lang;**

**public interface Comparable<E> {**

**public int compareTo(E o) { }**

**}**

1. package java.util;

public interface Comparable<E> {

public String compareTo(E o) { }

}

1. **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>()**

1. **No.** B.Yes
2. **Can a generic class have multiple generic parameters?**
3. **Yes**. B. No.
4. **Given int[] list = {1, 2, -1}, can you invoke sort(list) using the sort method in Listing 19.4?**
5. **No, because list is of type int[], but the sort method requires E[], where E is an object type.**
6. Yes,
7. **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?**
8. **No, because list is still of type int[], but the sort method requires E[], where E is an object type.**
9. Yes.
10. **What is a raw type?**
11. When you use generic type without specifying an actual parameter, it is called a raw type.
12. When you use generic type specifying an actual parameter, it is called a raw type.
13. None the above.
14. **Both A and B**
15. **Why is a raw type unsafe?**
16. **A raw type is unsafe, because some errors cannot be detected by the compiler.**
17. A raw type is not unsafe.
18. None of the above
19. Both A and B
20. **Why is the raw type allowed in Java?**
21. The raw type is not allowed in Java for backward compatibility.
22. **The raw type is allowed in Java for backward compatibility.**
23. The raw type is allowed in Java for removing compile error.
24. **What is the syntax to declare an ArrayList reference variable using the raw type and assign a raw type ArrayList object to it?**
25. **ArrayList list = new ArrayList();**
26. HashMap list=new List();
27. List list=new List();
28. Set list=new HashSet();
29. **Is GenericStack the same as GenericStack<Object>?**
30. **No, 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.**
31. Yes, it is same
32. None of the above.
33. **What are an unbounded wildcard, a bounded wildcard, and a lower-bound wildcard?**
34. ? is unbounded wildcard ,? super T is lower bounded wildcard ? extends T is bounded wildcard
35. **? is unbounded wildcard , ? extends T is bounded wildcard, ? super T is lower bounded wildcard**
36. ? extends T is bounded wildcard, ? super T is lower bounded wildcard, ? is unbounded wildcard ,
37. **If your program uses ArrayList<String> and ArrayList<Date> , does the JVM load both of them?**
38. **No. Only ArrayList is loaded.**
39. Yes, Both will load
40. **Can you create an instance using new E() for a generic type E? Why?**
41. **No, because the type information is not available at runtime.**
42. Yes

**19. Why are generics used?**  
a) Generics make code more fast  
b) Generics make code more optimised and readable  
**c) Generics add stability to your code by making more of your bugs detectable at compile time**d) Generics add stability to your code by making more of your bugs detectable at run time  
View Answer  
**Explanation**: Generics add stability to your code by making more of your bugs detectable at compile time.

**20. Which of these type parameters is used for a generic class to return and accept any type of object?**  
a) K  
b) N  
**c) T**  
d) V  
Explanation: T is used for type, A type variable can be any non-primitive type you specify: any class type, any interface type, any array type, or even another type variable..

**21. Which of these type parameters is used for a generic class to return and accept a number?**  
a) K  
**b) N**  
c) T  
d) V  
Explanation: N is used for Number.

**22. Which of these is an correct way of defining generic class?**  
a) class name(T1, T2, …, Tn) { /\* … \*/ }  
**b) class name { /\* … \*/ }**  
c) class name[T1, T2, …, Tn] { /\* … \*/ }  
d) class name{T1, T2, …, Tn} { /\* … \*/ }

23**. Which of the following is incorrect statement regarding the use of generics and parameterized types in Java?**  
a) Generics provide type safety by shifting more type checking responsibilities to the compiler  
b) Generics and parameterized types eliminate the need for down casts when using Java Collections  
**c) When designing your own collections class (say, a linked list), generics and parameterized types allow you to achieve type safety with just a single class definition as opposed to defining multiple classes**  
d) All of the mentioned

**24. Which of the following reference types cannot be generic?**  
**a) Anonymous inner class**  
b) Interface  
c) Inner class  
d) All of the mentioned

**Written Questions:**

**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).

**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.

**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.

**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).

**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

**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.

**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.

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

Because these methods cannot be implemented in the GenericMatric class.

**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.

**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.