Generics and Collections - 0

Given code of Test.java file:

```
class Printer<String> {
    private String t;
    Printer(String t){
        this.t = t;
    }
}

public class Test {
    public static void main(String[] args) {
        Printer<Integer> obj = new Printer<>(100);
        System.out.println(obj);
    }
}
```

What will be the result of compiling and executing Test class?

A - 100

B - Compilation error in Test class

C - Some text containing @ symbol

D - Compilation error in Printer class

Generics and Collections - 1

Given code of Test.java file:

```
import java.util.stream.IntStream;

public class Test {
    public static void main(String[] args) {
        IntStream stream = "OCP".chars();
        stream.forEach(c -> System.out.print((char)c));
        System.out.println(stream.count()); //Line 9
    }
}
```

What will be the result of compiling and executing Test class?

A - Runtime exception

B - None of the other options

C - Compilation error

D - OCP3

Generics and Collections - 2

Given code of Test.java file:

```
class T {
    @Override
    public String toString() {
        return "T";
    }
}

class Printer<T> {
    private T t;
    Printer(T t){
        this.t = t;
    }
    @Override
    public String toString(){
        return t.toString();
    }
}

public class Test {
    public static void main(String[] args) {
        Printer<T> obj = new Printer<>(new T());
        System.out.println(obj);
    }
}
```

What will be the result of compiling and executing Test class?

A - Compilation error in Printer class

B - Compilation error in Test class

C - Compilation error in T class

D - T

Generics and Collections - 3

Given code of Test.java file:

```
import java.util.*;

public class Test {
    public static void main(String[] args) {
        NavigableMap<Integer, String> map = new TreeMap<>();
        map.put(25, "Pune");
        map.put(32, "Mumbai");
        map.put(11, "Sri Nagar");
        map.put(39, "Chennai");

        System.out.println(map.headMap(25, true));
    }
}
```

A - {11=Sri Nagar, 25=Pune}

B - {32=Mumbai, 39=Chennai}

C - {11=Sri Nagar}

D - {25=Pune, 32=Mumbai, 39=Chennai}

Generics and Collections - 4

Given code of Test.java file:

```
import java.util.Arrays;

public class Test {
    public static void main(String[] args) {
        String [] cities = {"Seoul", "Tokyo", "Paris", "London",
                                "Hong Kong", "Singapore"};
        Arrays.stream(cities).sorted((s1,s2) -> s2.compareTo(s1))
                .forEach(System.out::println);
    }
}
```

What will be the result of compiling and executing Test class?

A -

```
Tokyo
Singapore
Seoul
Paris
London
Hong Kong
```

B -

```
Hong Kong
London
Paris
Seoul
Singapore
Tokyo
```

C - Compilation error

D -

```
Seoul
Tokyo
Paris
London
Hong Kong
Singapore
```

Generics and Collections - 5

Consider below code:

```
public class Test {
    public static void main(String[] args) {
        Operation o1 = (x, y) -> x + y;
        System.out.println(o1.operate(5, 10));
    }
}
```

Which of the following functional interface definitions can be used here, so that the output of above code is: 15? Select ALL that apply.

A -

```
interface Operation {
  long operate(long x, long y);
}
```

B -

```
interface Operation {
  int operate(int x, int y);
}
```

C -

```
interface Operation<T extends Integer> {
  T operate(T x, T y);
}
```

D -

```
interface Operation<T> {
  T operate(T x, T y);
}
```

Generics and Collections - 6

Given code of Test.java file:

```
import java.util.Map;
import java.util.TreeMap;

public class Test {
    public static void main(String[] args) throws Exception {
        Map<Integer, String> map = new TreeMap<>();
        map.put(1, "one");
        map.put(2, "two");
        map.put(3, "three");
        map.put(null, "null");
        map.forEach((key, value) -> System.out.println("{" + key +
        ": " + value + "}"));
    }
}
```

What will be the result of compiling and executing Test class?

A - NullPointerException is thrown at runtime

B -

```
{null: null}
{1: one}
{2: two}
{3: three}
```

C -

```
{1: one}
{2: two}
{3: three}
{null: null}
```

D -

```
{1: one}
{2: two}
{3: three}
```

Generics and Collections - 7

Given code of Test.java file:

```
import java.util.ArrayList;
import java.util.List;

public class Test {
    public static void main(String[] args) {
        List<? super String> list = new ArrayList<>();
        list.add("A");
        list.add("B");
        for(String str : list) {
            System.out.print(str);
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - Runtime exception

B - Compilation error

C - AB

Generics and Collections - 8

Given code of Test.java file:

```
import java.util.Arrays;
import java.util.List;
import java.util.ListIterator;

public class Test {
    public static void main(String[] args) {
        List<String> list = Arrays.asList("T", "S", "R", "I", "F");
        ListIterator<String> iter = list.listIterator(2);
        while(iter.hasNext()) {
            System.out.print(iter.next());
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - IF

B - Runtime Exception

C - RIF

Generics and Collections - 9

Given code of Test.java file:

```
import java.util.ArrayDeque;
import java.util.Deque;

public class Test {
    public static void main(String[] args) {
        Deque<Boolean> deque = new ArrayDeque<>();
        deque.push(new Boolean("abc"));
        deque.push(new Boolean("tRuE"));
        deque.push(new Boolean("FALSE"));
        deque.push(true);
        System.out.println(deque.pop() + ":" + deque.peek() + ":" +
         deque.size());
    }
}
```

What will be the result of compiling and executing Test class?

A - true:false:3

B - false:false:3

C - false:true:3

D - true:true:3

Generics and Collections - 10

Given code of Test.java file:

```
import java.util.Arrays;
import java.util.Comparator;
import java.util.List;

class Person {
    private String firstName;
    private String lastName;

    public Person(String firstName, String lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
    }

    public String getFirstName() {
        return firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public String toString() {
        return "{" + firstName + ", " + lastName + "}";
    }
}

public class Test {
    public static void main(String[] args) {
        List<Person> list = Arrays.asList(
                new Person("Tom", "Riddle"),
                new Person("Tom", "Hanks"),
                new Person("Yusuf", "Pathan"));

        list.stream().sorted(Comparator.comparing(Person::getFirstName).reverse

        .thenComparing(Person::getLastName)).forEach(System.out::println);
    }
}
```

What will be the result of compiling and executing Test class?

A -

```
{Tom, Hanks}
{Tom, Riddle}
{Yusuf, Pathan}
```

B -

```
{Yusuf, Pathan}
{Tom, Hanks}
{Tom, Riddle}
```

C -

{Yusuf, Pathan}
{Tom, Riddle}
{Tom, Hanks}

D -

{Tom, Riddle}
{Tom, Hanks}
{Yusuf, Pathan}

Generics and Collections - 11

Given code of Test.java file:

```
import java.util.*;

class Student {
    private String name;
    private int age;

    Student(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public int hashCode() {
        return name.hashCode() + age;
    }

    public String toString() {
        return "Student[" + name + ", " + age + "]";
    }

    public boolean equals(Object obj) {
        if(obj instanceof Student) {
            Student stud = (Student)obj;
            return this.name.equals(stud.name) && this.age ==
         stud.age;
        }
        return false;
    }

    public String getName() {return name;}

    public int getAge() {return age;}

    public static int compareByName(Student s1, Student s2) {
        return s1.getName().compareTo(s2.getName());
    }
}

public class Test {
    public static void main(String[] args) {
        Set<Student> students = new TreeSet<>
         (Student::compareByName);
        students.add(new Student("James", 20));
        students.add(new Student("James", 20));
        students.add(new Student("James", 22));

        System.out.println(students.size());
    }
}
```

What will be the result of compiling and executing Test class?

A - 3

B - 1

C - Runtime Exception

D - 2

Generics and Collections - 12

Given code of Test.java file:

```
package com.training.ocp;

class Animal {}

class Dog extends Animal {}

class Cat extends Animal {}

class A<T> {
    T t;
    void set(T t) {
        this.t = t;
    }

    T get() {
        return t;
    }
}

public class Test {
    public static <T> void print1(A<? extends Animal> obj) {
        obj.set(new Dog()); //Line 22
        System.out.println(obj.get().getClass());
    }

    public static <T> void print2(A<? super Dog> obj) {
        obj.set(new Dog()); //Line 27
        System.out.println(obj.get().getClass());
    }

    public static void main(String[] args) {
        A<Dog> obj = new A<>();
        print1(obj); //Line 33
        print2(obj); //Line 34
    }
}
```

What will be the result of compiling and executing Test class?

A - Runtime Exception

B -

```
null
class com.training.ocp.Dog
```

C -

```
class com.training.ocp.Dog
```

```
null
```

D -

```
class com.training.ocp.Dog
class com.training.ocp.Dog
```

E - Compilation error

Generics and Collections - 13

Given code of Test.java file:

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class Test {
    public static void main(String[] args) {
        List<Integer> list = new ArrayList<>
         (Arrays.asList(1,2,3,4,5,6,7,8,9,10));
        list.removeIf(i -> i % 2 == 1);
        System.out.println(list);
    }
}
```

What will be the result of compiling and executing Test class?

A - [1, 3, 5, 7, 9]

B - Compilation Error

C - Runtime Exception

D - [2, 4, 6, 8, 10]

Generics and Collections - 14

Given code of Test.java file:

```
import java.util.Arrays;
import java.util.Comparator;

public class Test {
    public static void main(String[] args) {
        String [] arr = {"A5", "B4", "C3", "D2", "E1"};
        Arrays.sort(arr, Comparator.comparing(s -> s.substring(1)));
        for(String str : arr) {
            System.out.print(str + " ");
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - E5 D4 C3 B2 A1

B - E1 D2 C3 B4 A5

C - A5 B4 C3 D2 E1

D - A1 B2 C3 D4 E5

Generics and Collections - 15

Given code of Test.java file:

```
import java.util.Arrays;
import java.util.List;

public class Test {
    public static void main(String[] args) {
        List<Integer> list = Arrays.asList(0,2,4,6,8);
        list.replaceAll(i -> i + 1);
        System.out.println(list);
    }
}
```

What will be the result of compiling and executing Test class?

A - [1, 3, 5, 7, 9]

B - Compilation error

C - [0, 2, 4, 6, 8]

D - Runtime Exception

Generics and Collections - 16

Given code of Test.java file:

```
import java.util.ArrayDeque;
import java.util.Deque;

public class Test {
    public static void main(String[] args) {
        Deque<Character> chars = new ArrayDeque<>();
        chars.add('A');
        chars.remove();
        chars.remove();

        System.out.println(chars);
    }
}
```

What will be the result of compiling and executing Test class?

A - []

B - Runtime Exception

C - [A]

Generics and Collections - 17

Given code of Test.java file:

```
import java.util.*;

enum TrafficLight {
    RED, YELLOW, GREEN
}

public class Test {
    public static void main(String[] args) {
        Map<TrafficLight, String> map = new TreeMap<>();
        map.put(TrafficLight.GREEN, "GO");
        map.put(TrafficLight.RED, "STOP");
        map.put(TrafficLight.YELLOW, "READY TO STOP");

        for(String msg : map.values()) {
            System.out.println(msg);
        }
    }
}
```

What will be the result of compiling and executing Test class?

A -

```
GO
STOP
READY TO STOP
```

B -

```
STOP
READY TO STOP
GO
```

C -

```
GO
READY TO STOP
STOP
```

D - Printing order cannot be predicted.

Generics and Collections - 18

Given code of Test.java file:

```
import java.util.Set;
import java.util.TreeSet;

class Employee implements Comparable<Employee> {
    private String name;
    private int age;

    Employee(String name, int age) {
        this.name = name;
        this.age = age;
    }

    @Override
    public String toString() {
        return "{" + name + ", " + age + "}";
    }

    @Override
    public int compareTo(Employee o) {
        return o.age - this.age;
    }
}

public class Test {
    public static void main(String[] args) {
        Set<Employee> employees = new TreeSet<>();
        employees.add(new Employee("Udayan", 31));
        employees.add(new Employee("Neha", 23));
        employees.add(new Employee("Hou Jian", 42));
        employees.add(new Employee("Smita", 29));

        System.out.println(employees);
    }
}
```

What will be the result of compiling and executing Test class?

A - [{Udayan, 31}, {Neha, 23}, {Hou Jian, 42}, {Smita, 29}]

B - [{Neha, 23}, {Smita, 29}, {Udayan, 31}, {Hou Jian, 42}]

C - [{Hou Jian, 42}, {Udayan, 31}, {Smita, 29}, {Neha, 23}]

D - Compilation error

Generics and Collections - 19

Given code of Test.java file:

```
import java.util.Arrays;
import java.util.List;

public class Test {
    public static void main(String[] args) {
        List<String> list = Arrays.asList("A", "E", "I", "O");
        list.add("U");
        list.forEach(System.out::print);
    }
}
```

What will be the result of compiling and executing Test class?

A - Runtime exception

B - Compilation error

C - UAEIO

D - AEIO

E - AEIOU

Generics and Collections - 20

Given code of Test.java file:

```
import java.util.ArrayList;
import java.util.List;

public class Test {
    public static void main(String[] args) {
        List<String> list1 = new ArrayList<>();
        list1.add("A");
        list1.add("B");

        List<? extends Object> list2 = list1;
        list2.remove("A"); //Line 13
        list2.add("C"); //Line 14

        System.out.println(list2);
    }
}
```

What will be the result of compiling and executing Test class?

A - Runtime exception

B - BC

C - ABC

D - Compilation error

Generics and Collections - 21

Given code of Test.java file:

```
public class Test<T> {
    private T t;

    public T get() {
        return t;
    }

    public void set(T t) {
        this.t = t;
    }

    public static void main(String args[]) {
        Test obj = new Test();
        obj.set("OCP");
        obj.set(85);
        obj.set('%');

        System.out.println(obj.get());
    }
}
```

What will be the result of compiling and executing Test class?

A - Runtime exception

B - Output contains some text containing @ symbol

C - %

D - OCP85%

E - Compilation error

Generics and Collections - 22

Given code of Test.java file:

```
import java.util.*;

public class Test {
    public static void main(String[] args) {
        Set<String> set = new TreeSet<>
         (Arrays.asList(null,null,null));
        long count = set.stream().count();
        System.out.println(count);
    }
}
```

What will be the result of compiling and executing Test class?

A - Runtime Exception

B - 0

C - 3

D - 1

Generics and Collections - 23

Given code of Test.java file:

```
class Printer<String> {
    private String t;

    Printer(String t){
        this.t = t;
    }

    public String toString() {
        return null;
    }
}

public class Test {
    public static void main(String[] args) {
        Printer<Integer> obj = new Printer<>(100);
        System.out.println(obj);
    }
}
```

What will be the result of compiling and executing Test class?

A - 100

B - null

C - Compilation error in Test class

D - Compilation error in Printer class

Generics and Collections - 24

Given code of Test.java file:

```
import java.util.Arrays;
import java.util.List;

public class Test {
    public static void main(String[] args) {
        List<String> list = Arrays.asList("7 Seven", "Lucky 7",
        "77", "07ne");
        list.stream().filter(str -> str.contains("7"))
                .forEach(System.out::println);
    }
}
```

What will be the result of compiling and executing Test class?

A - 7 Seven

B -

```
7 Seven
Lucky 7
77
07ne
```

C -

```
7 Seven
Lucky 7
```

D -

```
7 Seven
Lucky 7
77
```

E -

```
7 Seven
Lucky 7
07ne
```

Generics and Collections - 25

Consider below codes:

```
class A<T extends String> {

}

class B<T super String> {

}
```

Which of the following statement is correct?

A - Both class A and B compiles successfully

B - Only class B compiles successfully

C - Only class A compiles successfully

Generics and Collections - 26

Given code of Test.java file:

```
import java.util.Arrays;
import java.util.Collections;
import java.util.List;

class Name {
    String first;
    String last;

    public Name(String first, String last) {
        this.first = first;
        this.last = last;
    }

    public String getFirst() {
        return first;
    }

    public String getLast() {
        return last;
    }

    public String toString() {
        return first + " " + last;
    }

}

public class Test {
    public static void main(String[] args) {
        List<Name> names = Arrays.asList(new Name("Peter", "Lee"),
         new Name("John", "Smith"),
                new Name("bonita", "smith"));

        /*INSERT*/

        System.out.println(names);
    }
}
```

Currently on executing Test class, [Peter Lee, John Smith, bonita smith] is displayed in the output.

Which of the following options can replace /*INSERT*/ such that on executing Test class, [bonita smith, John Smith, Peter Lee] is displayed in the output? The names list must be sorted in ascending order of first name in case-insensitive manner. Select 3 options.

A -

```
Collections.sort(names, (01, 02) ->
        o1.getFirst().compareToIgnoreCase(o2.getFirst()));
```

B -

```
Collections.sort(names, (01, 02) ->
        o1.getFirst().toLowerCase().compareTo(o2.getFirst().toLowerCase()));
```

C -

```
Collections.sort(names, (01, 02) ->
        o1.getFirst().compareTo(o2.getFirst()));
```

D -

```
Collections.sort(names, (01, 02) ->
        o1.getFirst().toUpperCase().compareTo(o2.getFirst().touppercase()));
```

Generics and Collections - 27

Given code of Test.java file:

```
import java.util.stream.IntStream;

public class Test {
    public static void main(String[] args) {
        System.out.println(IntStream.range(10,1).count());
    }
}
```

What will be the result of compiling and executing Test class?

A - 9

B - 10

C - Runtime Exception

D - 0

Generics and Collections - 28

Given code of Test.java file:

```
import java.util.*;

enum TrafficLight {
    RED, YELLOW, GREEN
}

public class Test {
    public static void main(String[] args) {
        Map<TrafficLight, String> map = new TreeMap<>();
        map.put(TrafficLight.GREEN, "GO");
        map.put(TrafficLight.RED, "STOP");
        map.put(TrafficLight.YELLOW, "STOP IN 3 Seconds");
        map.put(TrafficLight.YELLOW, "READY TO STOP");

        for(String msg : map.values()) {
            System.out.println(msg);
        }
    }
}
```

What will be the result of compiling and executing Test class?

A -

```
STOP
STOP IN 3 Seconds
GO
```

B -

```
STOP
STOP IN 3 Seconds
READY TO STOP
GO
```

C -

```
STOP
READY TO STOP
GO
```

D - Printing order cannot be predicted.

E -

```
STOP
READY TO STOP
STOP IN 3 Seconds
GO
```

Generics and Collections - 29

Given code of Test.java file:

```
import java.util.*;

public class Test {
    public static void main(String[] args) {
        List<? extends String> list = new ArrayList<>
         (Arrays.asList("A", "E", "I", "O")); //Line 8
        list.add("U"); //Line 9
        list.forEach(System.out::print);
    }
}
```

What will be the result of compiling and executing Test class?

A - AEIO

B - AEIOU

C - Runtime exception

D - Line 8 causes compilation error

E - Line 9 causes compilation error

Generics and Collections - 30

Given code of Test.java file:

```
import java.util.Arrays;
import java.util.Comparator;
import java.util.List;

class Student implements Comparator<Student> {
    private String name;
    private String exam;

    public Student() {
        super();
    }

    public Student(String name, String exam) {
        this.name = name;
        this.exam = exam;
    }

    public int compare(Student s1, Student s2) {
        return s2.name.compareToIgnoreCase(s1.name);
    }

    public String toString() {
        return '{' + name + ", " + exam + '}';
    }
}

public class Test {
    public static void main(String[] args) {
        Student stud1 = new Student("John", "OCA");
        Student stud2 = new Student("Jack", "OCP");
        Student stud3 = new Student("Rob", "OCP");
        List<Student> list = Arrays.asList(stud1, stud2, stud3);
        list.sort(new Student());
        list.forEach(System.out::println);
    }
}
```

What will be the result of compiling and executing Test class?

A -

```
{Rob, OCP}
{John, OCA}
{Jack, OCP}
```

B -

```
{Jack, OCP}
{John, OCA}
```

```
{Rob, OCP}
```

C - Compilation error

D - Runtime exception

Generics and Collections - 31

Given code of Test.java file:

```
import java.util.Arrays;
import java.util.Collections;
import java.util.List;

public class Test {
    public static void main(String[] args) {
        List<String> list = Arrays.asList("M", "R", "A", "P");
        Collections.sort(list, null);
        list.stream().peek(System.out::print);
    }
}
```

What will be the result of compiling and executing Test class?

A - RPMA

B - MRAP

C - Runtime Exception

D - None of the other optionsAMPR

E - AMPR

Generics and Collections - 32

Given code of Test.java file:

```
public class Test {
    private static boolean isDirection(int ch) {
        switch(ch) {
            case 'N':
            case 'E':
            case 'W':
            case 'S':
                return true;
        }
        return false;
    }

    public static void main(String[] args) {
        String str = "North East West South";
        str.chars().filter(Test::isDirection)
                .forEach(c -> System.out.print((char)c));
    }
}
```

What will be the result of compiling and executing Test class?

A - orth ast est outh

B - None of the other options

C - N E W S

D - NEWS

Generics and Collections - 33

For the given code:

```
interface Operator<T> {
    public abstract T operation(T t1, T t2);
}

public class Test {
    public static void main(String[] args) {
        System.out.println(new Operator<String>() {
            public String operation(String s1, String s2) {
                return s1 + s2;
            }
        });
    }
}
```

Which of the following options successfully replace anonymous inner class code with lambda expression code?

A - System.out.println((si, s2) -> si + s2);

B - System.out.println((si, s2) -> { return si + s2; });

C - System.out.println((String si, String s2) -> si + s2);

D - None of the other options

Generics and Collections - 34

Given code of Test.java file:

```
public class Test<T> {
    static T obj;
}
```

Does above code compile successfully?

A - No

B - Yes

Generics and Collections - 35

For the code below:

```
import java.util.Arrays;

public class Test {
    public static void main(String[] args) {
        String [] arr = {"**", "***", "*", "******", "****"};
        Arrays.sort(arr, (s1, s2) -> s1.length()-s2.length());
        for(String str : arr) {
            System.out.println(str);
        }
    }
}
```

What do you need to do so that above code gives following output?

```
*
**
***
****
*****
```

A - Add the import statement for the Comparator interface: import java.util.Comparator;

B - Change the lambda expression to (s2, si) -> s1.length()-s2.length()

C - Change the lambda expression to (si, s2) -> s2.length()-si.length()

D - Existing code without any changes displays above output.

Generics and Collections - 36

Given code of Test.java file:

```
import java.util.ArrayDeque;
import java.util.Arrays;
import java.util.Deque;
import java.util.List;

public class Test {
    public static void main(String[] args) throws Exception {
        List<String> list = Arrays.asList("oca", null, "ocp",
        "java", "null"); //Line n1
        Deque<String> deque = new ArrayDeque<String>(list); //Line
        n2
        System.out.println(deque.size()); //Line n3
    }
}
```

What will be the result of compiling and executing Test class?

A - 4

B - 5

C - NullPointerException is thrown at runtime

D - 3

Generics and Collections - 37

Given code of Test.java file:

```
import java.util.Arrays;
import java.util.List;
import java.util.ListIterator;

public class Test {
    public static void main(String[] args) {
        List<String> list = Arrays.asList("T", "S", "R", "I", "F");
        ListIterator<String> iter = list.listIterator(5);
        while(iter.hasPrevious()) {
            System.out.print(iter.previous());
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - IRST

B - FIRST

C - Runtime Exception

D - TSRIF

Generics and Collections - 38

Given code of Test.java file:

```
import java.util.Arrays;
import java.util.Collections;
import java.util.List;

public class Test {
    public static void main(String[] args) {
        List<String> list = Arrays.asList("M", "R", "A", "P");
        Collections.sort(list, null);
        System.out.println(list);
    }
}
```

What will be the result of compiling and executing Test class?

A - [A, M, P, R]

B - [R, P, M, A]

C - Runtime Exception

D - [M, R, A, P]

Generics and Collections - 39

Given code of Test.java file:

```
import java.util.LinkedList;
import java.util.List;
import java.util.Queue;

public class Test {
    public static void main(String[] args) {
        List<String> list = new LinkedList<>();
        list.add("ONE");
        list.add("TWO");
        list.remove(1);
        System.out.println(list);

        Queue<String> queue = new LinkedList<>();
        queue.add("ONE");
        queue.add("TWO");
        queue.remove();
        System.out.println(queue);
    }
}
```

What will be the result of compiling and executing Test class?

A -

```
[ONE]
[ONE]
```

B -

```
[ONE]
[TWO]
```

C -

```
[TWO]
[TWO]
```

D -

```
[TWO]
[ONE]
```

Generics and Collections - 40

Given code of Test.java file:

```
import java.util.Arrays;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;

public class Test {
    public static void main(String[] args) {
        List<String> emails = Arrays.asList("training@outlook.com",
        "ocpjp@outlook.com", "ocpjp@gmail.com",
                "training@gmail.com");
        Collections.sort(emails, Comparator.comparing(str ->
         str.substring(str.indexOf("@") + 1)));
        for(String email : emails) {
            System.out.println(email);
        }
    }
}
```

What will be the result of compiling and executing Test class?

A -

```
ocpjp@gmail.com
training@gmail.com
ocpjp@outlook.com
training@outlook.com
```

B -

```
ocpjp@outlook.com
training@outlook.com
training@gmail.com
ocpjp@gmail.com
```

C -

```
ocpjp@outlook.com
training@outlook.com
ocpjp@gmail.com
training@gmail.com
```

D -

```
ocpjp@gmail.com
training@gmail.com
training@outlook.com
ocpjp@outlook.com
```

Generics and Collections - 41

Does below code compile successfully?

```
class GenericPrinter<T> {}
abstract class AbstractGenericPrinter<X,Y,T> extends
        GenericPrinter<T>{}
```

A - No

B - Yes

Generics and Collections - 42

Given code of Test.java file:

```
class A{}
interface M{}
interface N{}

class B extends A {}
class C extends A implements M {}
class D extends A implements M, N {}

class Generic<T extends A & M & N> {}

public class Test {
    public static void main(String[] args) {
        /*INSERT*/
    }
}
```

Which of the following statements, if used to replace /*INSERT*/, will not cause any compilation error?

A - All options will work

B - Generic obj = new Generic<>();

C - Generic obj = new Generic<>();

D - Generic obj = new Generic<>();

E - Generic obj = new Generic<>();

Generics and Collections - 43

Given code of Test.java file:

```
import java.util.ArrayList;
import java.util.List;

public class Test {
    public static void main(String[] args) {
        List list = new ArrayList<String>();
        list.add(1);
        list.add("2");
        list.forEach(System.out::print);
    }
}
```

Which of the following is correct?

Select 2 options.

A - 12 is displayed on to the console

B - Code compiles with some warnings

C - Code compiles without any errors and warnings

D - Exception is thrown at runtime

Generics and Collections - 44

Given code of Test.java file:

```
public class Test {
    private <T extends Number> static void print(T t) {
        System.out.println(t.intValue());
    }

    public static void main(String[] args) {
        print(new Double(5.5));
    }
}
```

What will be the result of compiling and executing Test class?

A - 5

B - Runtime Exception

C - Compilation error

D - 6

Generics and Collections - 45

Given code of Test.java file:

```
import java.util.*;

public class Test {
    public static void main(String[] args) {
        Set<Character> set = new TreeSet<>
         (Arrays.asList('a','b','c','A','a','c'));
        set.stream().forEach(System.out::print);
    }
}
```

What will be the result of compiling and executing Test class?

A - abc

B - Aabc

C - abcAac

D - Aaabcc

Generics and Collections - 46

What will be the result of compiling and executing Test class?

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;

class Point {
    private int x;
    private int y;

    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public int getX() {
        return x;
    }

    public int getY() {
        return y;
    }

    @Override
    public String toString() {
        return "Point(" + x + ", " + y + ")";
    }
}

public class Test {
    public static void main(String [] args) {
        List<Point> points = new ArrayList<>();
        points.add(new Point(4, 5));
        points.add(new Point(6, 7));
        points.add(new Point(2, 2));

        Collections.sort(points, new Comparator<Point>() {
            public int compare(Point o1, Point o2) {
                return o2.getX() - o1.getX();
            }
        });

        System.out.println(points);
    }
}
```

A - [Point(2, 2), Point(4, 5), Point(6, 7)]

B - [Point(6, 7), Point(4, 5), Point(2, 2)]

C - [Point(4, 5), Point(6, 7), Point(2, 2)]

D - Compilation error

Generics and Collections - 47

Given code of Test.java file:

```
class Printer<T implements Cloneable> {}

public class Test {
    public static void main(String[] args) {
        Printer<String> printer = new Printer<>();
        System.out.println(printer);
    }
}
```

What will be the result of compiling and executing Test class?

A - Compilation error for Printer class

B - Some text containing @ symbol

C - Compilation error for Test class

Generics and Collections - 48

Given code of Test.java file:

```
import java.util.*;

public class Test {
    public static void main(String[] args) {
        NavigableSet<String> set = new TreeSet<>
         (Arrays.asList("red", "green", blue", "gray"));
        System.out.println(set.ceiling("gray"));
        System.out.println(set.floor("gray"));
        System.out.println(set.higher("gray"));
        System.out.println(set.lower("gray"));
    }
}
```

What will be the result of compiling and executing Test class?

A -

```
gray
gray
green
blue
```

B -

```
green
blue
gray
gray
```

C -

```
gray
gray
gray
gray
```

D -

```
green
blue
green
blue
```

Generics and Collections - 49

Given code of Test.java file:

```
import java.util.*;

public class Test {
    public static void main(String[] args) {
        List<String> colors = new ArrayList<>();
        colors.add("RED");
        colors.add("GREEN");
        colors.add("BLUE");
        Iterator<String> iter = colors.iterator();
        while(iter.hasNext()) {
            iter.remove();
            iter.next();
        }
        System.out.println(colors.size());
    }
}
```

What will be the result of compiling and executing Test class?

A - Runtime exception

B - 2

C - 0

Generics and Collections - 50

Given code of Test.java file:

```
import java.util.*;

public class Test {
    public static void main(String[] args) {
        Map<Integer, String> map = new LinkedHashMap<>();
        map.put(null, "zero");
        map.put(1, "one");

        System.out.println(map);
    }
}
```

What will be the result of compiling and executing Test class?

A - {null=zero, 1=one}

B - {1=one, null=zero}

C - Runtime Exception

D - Order cannot be predicted

Generics and Collections - 51

Given code of Test.java file:

```
import java.util.*;

public class Test {
    public static void main(String[] args) {
        int i = 2000;
        Deque<Integer> deque = new ArrayDeque<>();
        deque.add(1000);
        deque.add(i);
        deque.add(3000);

        /*INSERT*/
    }
}
```

Which of the following statements, if used to replace /*INSERT*/, will print following on to the console:

1000 2000 3000

A - deque.forEach(i -> System.out.println(i));

B - deque.forEach(s -> System.out.println(s));

C - deque.forEach(System.out::println);

D - deque.forEach(System.out::print);

Generics and Collections - 52

Consider below code:

```
import java.util.Arrays;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;

public class Test {
    public static void main(String [] args) {
        List<String> names = Arrays.asList("James", "diana",
        "Anna");

        /*INSERT*/

        System.out.println(names);
    }
}
```

Currently on executing Test class, [James, diana, Anna] is printed in the output.

Which of the following options can replace /*INSERT*/ such that on executing Test class, [Anna, diana, James] is printed in the output?

A - Collections.sort(names);

B -

```
Collections.sort(names, new Comparator<String>() {
  public int compare(String 01, String 02) {
    return 01.compareTo(02);
  }
});
```

C -

```
Collections.sort(names, new Comparator<String>() {
  public int compare(String 01, String 02) {
    return 02.compareTo(o1);
  }
});
```

D -

```
Collections.sort(names, new Comparator<String>() {
  public int compare(String 01, String 02) {
    return 0o1.compareToIgnoreCase(o2);
  }
});
```

Generics and Collections - 53

Given code of Test.java file:

```
import java.util.*;

class Employee {
    private String name;
    private double salary;

    public Employee(String name, double salary) {
        this.name = name;
        this.salary = salary;
    }

    public String getName() {
        return name;
    }

    public double getSalary() {
        return salary;
    }

    public void setSalary(double salary) {
        this.salary = salary;
    }

    public String toString() {
        return "{" + name + ", " + salary + "}";
    }
}

public class Test {
    public static void main(String[] args) {
        List<Employee> employees = Arrays.asList(new
         Employee("Jack", 10000),
                new Employee("Lucy", 12000));
        employees.stream().peek(e -> e.setSalary(e.getSalary() +
         1000))
                .forEach(System.out::println);
    }
}
```

What will be the result of compiling and executing Test class?

A -

```
{Lucy, 12000.0}
{Jack, 10000.0}
```

B -

```
{Lucy, 13000.0}
```

{Jack, 11000.0}

C -

{Jack, 10000.0}
{Lucy, 12000.0}

D -

{Jack, 11000.0}
{Lucy, 13000.0}

Generics and Collections - 54

What will be the result of compiling and executing TestPoint class?

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;

class Point {
    private int x;
    private int y;

    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

    @Override
    public String toString() {
        return "Point(" + x + ", " + y + ")";
    }
}

public class TestPoint {
    public static void main(String [] args) {
        List<Point> points = new ArrayList<>();
        points.add(new Point(4, 5));
        points.add(new Point(6, 7));
        points.add(new Point(2, 2));

        Collections.sort(points, new Comparator<Point>() {
            @Override
            public int compare(Point o1, Point o2) {
                return o1.x - o2.x;
            }
        });
    }
}
```

A - [Point(6, 7), Point(4, 5), Point(2, 2)]

B - [Point(2, 2), Point(4, 5), Point(6, 7)]

C - Compilation error

D - [Point(4, 5), Point(6, 7), Point(2, 2)]

Generics and Collections - 55

Given code of Test.java file:

```
import java.util.*;

class Employee {
    private String name;
    private double salary;

    public Employee(String name, double salary) {
        this.name = name;
        this.salary = salary;
    }

    public String getName() {
        return name;
    }

    public double getSalary() {
        return salary;
    }

    public void setSalary(double salary) {
        this.salary = salary;
    }

    public String toString() {
        return "{" + name + ", " + salary + "}";
    }
}

public class Test {
    public static void main(String[] args) {
        List<Employee> employees = Arrays.asList(new
         Employee("Jack", 10000),
                new Employee("Lucy", 12000));
        employees.stream().filter(x -> x.getSalary() > 10000)
                .map(e -> e.getName()).forEach(System.out::println);
    }
}
```

What will be the result of compiling and executing Test class?

A - Lucy

B -

```
Jack
Lucy
```

C - Jack

D-

Lucy
Jack

Generics and Collections - 56

Given code of Test.java file:

```
import java.util.Arrays;
import java.util.List;

public class Test {
    public static void main(String[] args) {
        List<String> list = Arrays.asList("A", "A", "b", "B", "c",
        "c");
        list.stream().distinct().forEach(System.out::print);
    }
}
```

What will be the result of compiling and executing Test class?

A - ABc

B - ABbc

C - Abc

D - AbBc

E - AAbBcc

Generics and Collections - 57

Given code of Test.java file:

```
import java.util.ArrayDeque;
import java.util.Deque;

public class Test {
    public static void main(String[] args) {
        Deque<Character> chars = new ArrayDeque<>();
        chars.add('A');
        chars.add('B');
        chars.remove();
        chars.add('C');
        chars.remove();

        System.out.println(chars);
    }
}
```

What will be the result of compiling and executing Test class?

A - [C]

B - [B]

C - [A]

Generics and Collections - 58

Given code of Test.java file:

```
import java.util.stream.IntStream;

public class Test {
    public static void main(String[] args) {
        System.out.println(IntStream.range(-10, -10).count());
        System.out.println(IntStream.rangeClosed(-10, -10).count());
    }
}
```

What will be the result of compiling and executing Test class?

A -

```
1
1
```

B -

```
1
0
```

C -

```
0
0
```

D -

```
0
1
```

Generics and Collections - 59

Does below code compile successfully?

```
class A{}
interface M{}
interface N{}

class B extends A {}
class C extends A implements M {}
class D extends A implements M, N {}

class Generic<T extends M & N & A> {}
```

A - No

B - Yes

Generics and Collections - 60

A bank's swift code is generally of 11 characters and used in international money transfers An example: ICICINBBRT4 ICIC: First 4 letters for bank code IN: Next 2 letters for Country code BB: Next 2 letters for Location code RT4: Next 3 letters for Branch code

Given code of Test.java file:

```
import java.util.Arrays;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;

public class SortSwiftCode {
    public static void main(String[] args) {
        List<String> swiftCodes = Arrays.asList("ICICINDD016",
         "ICICINBBRT4", "BOTKINDD075", "BARBINBB011",
                "SBBJINDD062", "ABNATHBK865", "BKCHTHBK012");

        Comparator<String> countryLocationBank =
         Comparator.comparing(SortSwiftCode::extractCountry)

         .thenComparing(SortSwiftCode::extractLocation).thenComparing(SortSwiftCode::ex

        Collections.sort(swiftCodes, countryLocationBank);
        printCodes(swiftCodes);

    }

    private static String extractCountry(String swiftCode) {
        return swiftCode.substring(4, 6);
    }

    private static String extractLocation(String swiftCode) {
        return swiftCode.substring(6, 8);
    }

    private static String extractBank(String swiftCode) {
        return swiftCode.substring(0, 4);
    }

    private static void printCodes(List<String> list) {
        for (String str : list) {
            System.out.println(str);
        }
    }
}
```

What will be the result of compiling and executing SortSwiftCode class?

A -

```
ABNATHBK865
BKCHTHBK012
BARBINBB011
ICICINBBRT4
BOTKINDD075
ICICINDD016
SBBJINDD062
```

B - None of the other options

C -

```
BARBINBB011
ICICINBBRT4
BOTKINDD075
```

ICICINDDO16
SBBJINDDO62
ABNATHBK8&65
BKCHTHBKO12

D -

BARBINBBO11
BOTKINDDO75
ICICINBBRT4
ICICINDDO16
SBBJINDDO62
ABNATHBK8&65
BKCHTHBKO12

Generics and Collections - 61

Given code of Test.java file:

```
import java.util.stream.IntStream;

public class Test {
    public static void main(String[] args) {
        IntStream.iterate(1, i -> i + 1).limit(11)
                .filter(i -> i % 2 != 0).forEach(System.out::print);
    }
}
```

What will be the result of compiling and executing Test class?

A - 13579

B - 1357911

C - 246810

D - 24681012

Generics and Collections - 62

Given code of Test.java file:

```
public class Test {
    private static <T extends Number> void print(T t) {
        System.out.println(t.intValue());
    }

    public static void main(String[] args) {
        /*INSERT*/
    }
}
```

Which of the following statements, if used to replace /*INSERT*/, will not cause any compilation error?

A - print(new Double(5.5));

B - print(new Integer(1));

C - print(new Object());

D - print(new Number(0));

E - print(new Character('a'));

Generics and Collections - 63

Given code of Test.java file:

```
import java.util.*;

class Student {
    private String name;
    private int age;

    Student(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public String toString() {
        return "Student[" + name + ", " + age + "]";
    }

    public boolean equals(Object obj) {
        if(obj instanceof Student) {
            Student stud = (Student)obj;
            return this.name.equals(stud.name) && this.age ==
         stud.age;
        }
        return false;
    }
}

public class Test {
    public static void main(String[] args) {
        Set<Student> students = new HashSet<>();
        students.add(new Student("James", 20));
        students.add(new Student("James", 20));
        students.add(new Student("James", 22));

        System.out.println(students.size());
    }
}
```

What will be the result of compiling and executing Test class?

A - 2

B - Runtime Exception

C - 3

Generics and Collections - 64

What will be the result of compiling and executing Test class?

```
interface Operator<T> {
    public abstract T operation(T t1, T t2);
}

public class Test {
    public static void main(String[] args) {
        Operator<String> opr1 = (s1, s2) -> s1 + s2;
        Operator<Integer> opr2 = (i1, i2) -> i1 + i2;
        opr1.operation("Hello", "World");
        opr2.operation(10, 40);
    }
}
```

A -

```
HelloWorld
50
```

B - Compilation error

C -

```
HelloWorld
1040
```

D - Program compiles and executes successfully but nothing is printed on to the console

Generics and Collections - 65

Given code of Test.java file:

```
public class Test {
    public static <T> T get(T t) {
        return t;
    }

    public static void main(String[] args) {
        String str = get("HELLO");
        System.out.println(str);
    }
}
```

What will be the result of compiling and executing Test class?

A - Compilation error in ‘main’ method

B - HELLO

C - Runtime Exception

D - Compilation error in ‘get’ method

Generics and Collections - 66

Given code of Test.java file:

```
import java.util.*;

public class Test {
    public static void main(String[] args) {
        NavigableMap<Integer, String> map = new TreeMap<>();
        map.put(25, "Pune");
        map.put(32, "Mumbai");
        map.put(11, "Sri Nagar");
        map.put(39, "Chennai");

        System.out.println(map.headMap(25));
        System.out.println(map.tailMap(25));
    }
}
```

What will be the result of compiling and executing Test class?

A -

```
{11=Sri Nagar}
{32=Mumbai, 39=Chennai}
```

B -

```
{11=Sri Nagar}
{25=Pune, 32=Mumbai, 39=Chennai}
```

C -

```
{11=Sri Nagar, 25=Pune}
{25=Pune, 32=Mumbai, 39=Chennai}
```

D -

```
{11=Sri Nagar, 25=Pune}
{32=Mumbai, 39=Chennai}
```

Generics and Collections - 67

Given code of Test.java file:

```
import java.util.*;

public class Test {
    public static void main(String[] args) {
        Deque<Integer> deque = new ArrayDeque<>();
        deque.add(100);
        deque.add(200);
        deque.addFirst(300);
        deque.addLast(400);
        deque.remove(200);

        System.out.println(deque.getFirst());
    }
}
```

What will be the result of compiling and executing Test class?

A - 300

B - 100

C - 400

D - 200

Generics and Collections - 68

What will be the result of compiling and executing Test class?

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;

class Point {
    private int x;
    private int y;

    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public int getX() {
        return x;
    }

    public int getY() {
        return y;
    }

    @Override
    public String toString() {
        return "Point(" + x + ", " + y + ")";
    }
}

public class Test {
    public static void main(String [] args) {
        List<Point> points = new ArrayList<>();
        points.add(new Point(4, 5));
        points.add(new Point(6, 7));
        points.add(new Point(2, 2));

        Collections.sort(points, new Comparator<Point>() {
            public int compareTo(Point o1, Point o2) {
                return o1.getX() - o2.getX();
            }
        });

        System.out.println(points);
    }
}
```

A - [Point(4, 5), Point(6, 7), Point(2, 2)]

B - Compilation error

C - [Point(6, 7), Point(4, 5), Point(2, 2)]

D - [Point(2, 2), Point(4, 5), Point(6, 7)]

Generics and Collections - 69

Given code of Test.java file:

```
class Printer<T extends String> {}

public class Test {
    public static void main(String[] args) {
        Printer<String> printer = new Printer<>();
        System.out.println(printer);
    }
}
```

What will be the result of compiling and executing Test class?

A - Compilation error for Test class

B - Some text containing @ symbol

C - Compilation error for Printer class

Generics and Collections - 70

Given code of Test.java file:

```
import java.util.Arrays;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;

public class Test {
    public static void main(String[] args) {
        List<String> list = Arrays.asList("#####", "#", "##",
        "####", "###");
        Comparator<String> comp = Comparator.comparing(s -> s);
        Collections.sort(list, comp.reversed());
        printCodes(list);

    }

    private static void printCodes(List<String> list) {
        for (String str : list) {
            System.out.println(str);
        }
    }
}
```

What will be the result of compiling and executing Test class?

A -

```
#####
#
##
####
###
```

B -

```
#####
####
###
##
#
```

C -

```
###
####
##
#
#####
```

D -

```
#
##
###
####
#####
```

Generics and Collections - 71

Given code of Test.java file:

```
public class Test<T> {
    T [] obj;

    public Test() {
        obj = new T[100];
    }

    public T [] get() {
        return obj;
    }

    public static void main(String[] args) {
        Test<String> test = new Test<>();
        String [] arr = test.get();
        System.out.println(arr.length);
    }
}
```

What will be the result of compiling and executing Test class?

A - Runtime exception

B - Compilation error

C - 100

Generics and Collections - 72

Given code of Test.java file:

```
import java.util.ArrayList;
import java.util.List;

public class Test {
    public static void main(String[] args) {
        List list = new ArrayList<Integer>();
        list.add(1);
        list.add(2);
        list.add("3"); //Line 11
        list.removeIf(i -> i % 2 == 1); //Line 12
        System.out.println(list);
    }
}
```

What will be the result of compiling and executing Test class?

A - Runtime Exception

B - Compilation error at Line 12

C - Compilation error at Line 11

D - [2]

Generics and Collections - 73

Given code of Test.java file:

```
import java.util.*;

class Student {
    private String name;
    private int age;

    Student(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public int hashCode() {
        return name.hashCode() + age;
    }

    public String toString() {
        return "Student[" + name + ", " + age + "]";
    }

    public boolean equals(Object obj) {
        if(obj instanceof Student) {
            Student stud = (Student)obj;
            return this.name.equals(stud.name) && this.age ==
         stud.age;
        }
        return false;
    }
}

public class Test {
    public static void main(String[] args) {
        Set<Student> students = new TreeSet<>();
        students.add(new Student("James", 20));
        students.add(new Student("James", 20));
        students.add(new Student("James", 22));

        System.out.println(students.size());
    }
}
```

What will be the result of compiling and executing Test class?

A - Runtime Exception

B - 2

C - 3

Generics and Collections - 74

Given code of Test.java file:

```
public class Test {
    private static final <X extends Integer, Y extends Integer> void
        add(X x, Y y) {
        System.out.println(x + y);
    }

    public static void main(String[] args) {
        add(10, 20);
    }
}
```

What will be the result of compiling and executing Test class?

A - 1020

B - Compilation error

C - Runtime Exception

D - 30

Generics and Collections - 75

Given code of Test.java file:

```
import java.util.ArrayList;
import java.util.List;

public class Test {
    public static void main(String[] args) {
        List<StringBuilder> list = new ArrayList<>();
        list.add(new StringBuilder("abc"));
        list.add(new StringBuilder("xyz"));
        list.stream().map(x -> x.reverse());
        System.out.println(list);
    }
}
```

What will be the result of compiling and executing Test class?

A - Compilation error

B - [cba, zyx]

C - Runtime Exception

D - [abc, xyz]

Generics and Collections - 76

Given code of Test.java file:

```
import java.util.stream.LongStream;

public class Test {
    public static void main(String[] args) {
        LongStream.iterate(0, i -> i +
         2).limit(4).forEach(System.out::print);
    }
}
```

What will be the result of compiling and executing Test class?

A - 0246

B - 02468

C - 02

D - 024

Generics and Collections - 77

Given code of Test.java file:

```
import java.util.*;
import java.util.stream.IntStream;

public class Test {
    public static void main(String[] args) {
        IntStream.range(1, 10).forEach(System.out::print);
    }
}
```

What will be the result of compiling and executing Test class?

A - 246810

B - 13579

C - 123456789

D - 12345678910

Generics and Collections - 78

Given code of Test.java file:

```
import java.util.*;

class Employee {
    private String name;
    private double salary;

    public Employee(String name, double salary) {
        this.name = name;
        this.salary = salary;
    }

    public String getName() {
        return name;
    }

    public double getSalary() {
        return salary;
    }

    public void setSalary(double salary) {
        this.salary = salary;
    }

    public String toString() {
        return "{" + name + ", " + salary + "}";
    }
}

public class Test {
    public static void main(String[] args) {
        List<Employee> employees = Arrays.asList(new
         Employee("Jack", 10000), new Employee("Lucy", 12000));
        employees.forEach(e -> e.setSalary(e.getSalary() +
         (e.getSalary() * .2)));
        employees.forEach(System.out::println);
    }
}
```

What will be the result of compiling and executing Test class?

A -

```
{Jack, 10000}
{Lucy, 12000}
```

B -

```
{Jack, 12000.0}
{Lucy, 14400.0}
```

C -

```
{Jack, 12000}
{Lucy, 14400}
```

D -

```
{Jack, 10000.0}
{Lucy, 12000.0}
```

Generics and Collections - 79

Given code of Test.java file:

```
import java.util.ArrayList;
import java.util.List;

abstract class Animal {}
class Dog extends Animal{}

public class Test {
    public static void main(String [] args) {
        List<Animal> list = new ArrayList<Dog>();
        list.add(0, new Dog());
        System.out.println(list.size() > 0);
    }
}
```

What will be the result of compiling and executing Test class?

A - Compilation error

B - Runtime exception

C - false

D - true

Generics and Collections - 80

Given code of Test.java file:

```
import java.util.*;

public class Test {
    public static void main(String[] args) {
        Set<String> set = new HashSet<>
         (Arrays.asList(null,null,null));
        long count = set.stream().count();
        System.out.println(count);
    }
}
```

What will be the result of compiling and executing Test class?

A - 0

B - 1

C - Runtime exception

D - 3