Μεθοδολογία Προγραμματισμού

Εισαγωγή στη Java - Java interfaces/abstract classes

1. (1 Μονάδες) Δείτε τον ακόλουθο κώδικα. Τι θα τυπώσει;

```
public class Person {
    private String name;

    public Person(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }
}

public class Main {
    public static void main(String[] args) {
      Set<Person> persons = new HashSet<>();

      Person person1 = new Person("Nikos");
      Person person2 = new Person("Nikos");

      if (person1.equals(person2)) {
          System.out.println("They␣are␣the␣same");
      } else {
          System.out.println("They␣are␣not␣the␣same");
      }

      persons.add(person1);

      if (persons.contains(person2)) {
          System.out.println("person2␣is␣contained␣in␣persons");
      } else {
          System.out.println("person2␣is␣not␣contained␣in␣persons");
      }
    }
}
```

Α′. They are the same
person2 is not contained in persons

Β′. They are not the same
person2 is not contained in persons

Γ′. They are the same
person2 is contained in persons

Δ′. They are not the same
person2 is contained in persons

2. (1 Μονάδες) Έστω ότι αλλάζαμε τον κώδικα ως εξής:

```
public class Person {
    private String name;

    public Person(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Person)) return false;
        Person person = (Person) o;
        return Objects.equals(getName(), person.getName());
    }
}
```

Αν το τρέχαμε θα τύπωνε:

Α΄. They are the same
person2 is not contained in persons

Β΄. They are not the same
person2 is not contained in persons

Γ΄. They are the same
person2 is contained in persons

Δ΄. They are not the same
person2 is contained in persons

3. (1 Μονάδες) Έστω ότι αλλάζαμε τον κώδικα ως εξής:

```
public class Person {
    private String name;

    public Person(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Person)) return false;
        Person person = (Person) o;
        return Objects.equals(getName(), person.getName());
    }

    @Override
    public int hashCode() {

        return Objects.hash(getName());
    }
}
```

A′. They are the same
person2 is not contained in persons

B′. They are not the same
person2 is not contained in persons

Γ′. They are the same
person2 is contained in persons

Δ′. They are not the same
person2 is contained in persons

4. (1 Μονάδες) Κοιτάξτε τον παρακάτω κώδικα, που προσπαθεί να αφαιρέσει ένα στοιχείο από μια λίστα με τρεις διαφορετικούς τρόπους.

```
1  public class Main {
2      public static void main(String[] args) {
3        List<String> strings = new ArrayList<>();
4
5        strings.add("One");
6        strings.add("Two");
7        strings.add("Three");
8        strings.add("Four");
9
10       strings.removeIf(x -> x.equals("Two"));
11
12       System.out.println(strings);
13
14
15       for (Iterator<String> stringIterator = strings.iterator();stringIterator.
             hasNext();) {
16           String item = stringIterator.next();
17           if (item.equals("One")) {
18               stringIterator.remove();
19           }
20       }
21       System.out.println(strings);
22
23
24         for ( Iterator<String> stringIterator = strings.iterator();stringIterator.
               hasNext();) {
25             String item = stringIterator.next();
26             if (item.equals("Four")) {
27                 strings.remove("Four");
28             }
29         }
30         System.out.println(strings);
31
32     }
33 }
```

Αν προσπαθούσαμε να ξαναορίσουμε τη μέθοδο doStatic() στην κλάση όπως κάναμε με τη doThisLater()

Α΄. Και οι τρεις τρόποι είναι σωστοί

Β΄. Η μέθοδος στις γραμμές 14-20 θα βγάλει το μήνυμα

Exception in thread "main" java.util.ConcurrentModificationException

Γ΄. Η μέθοδος στις γραμμές 24-29 θα βγάλει το μήνυμα

Exception in thread "main" java.util.ConcurrentModificationException

Δ΄. Καμία μέθοδος δεν είναι σωστή.

5. (1 Μονάδες) Για την αρχικοποίηση μιας λίστας θα μπορούσαμε να χρησιμοποιήσουμε τη μέθοδο Arrays.asList(), το javadoc της οποίας είναι αυτό:

   @SafeVarargs
   public static <T> List<T> asList(T... a)
   Returns a fixed-size list backed by the specified array.
   (Changes to the returned list "write through" to the array.)

   Parameters:
   a - the array by which the list will be backed
   Returns:
   a list view of the specified array

   Ποια από τις παρακάτω γραμμές θα πετύχαινε την αρχικοποίηση του προηγούμενου παραδείγματος;

```
1 strings = Arrays.asList(new String[]{"One", "Two", "Three", "Four"});
2 strings = Arrays.asList("One", "Two", "Three", "Four");
```

   Α΄. Καμία

   Β΄. Η δεύτερη

   Γ΄. Και οι δύο

   Δ΄. Η πρώτη

6. (1 Μονάδες) Υλοποιήστε μια στοίβα χρησιμοποιώντας ένα ArrayList().

7. (1 Μονάδες) Δύο κοινές υλοποιήσεις μιας λίστας List στη Java είναι η ArrayList και η LinkedList Τι από τα παρακάτω είναι σωστά;

   Α΄. Αν συνήθως παίρνουμε στοιχεία από συγκεκριμένες θέσεις get(index) είναι γρηγορότερη η LinkedList, ενώ αν αφαιρούμε η ArrayList

   Β΄. Αν συνήθως προσθέτουμε στοιχεία στην αρχή add(0,element) είναι παρόμοια η LinkedList, με τη ArrayList

   Γ΄. Αν συνήθως παίρνουμε στοιχεία από συγκεκριμένες θέσεις get(index) είναι γρηγορότερη η ArrayList, ενώ αν αφαιρούμε η LinkedList

   Δ΄. Η (β) και (γ)

   Ε΄. Καμία απάντηση δεν είναι σωστή