INFORMACIJA, LOGIKA I JEZICI

završni ispit - ogledni primjerak

Zadatak 1. Koliko je od navedenih tvrdnji pogrešno?

a) Java i C++ su objektno orijentirani programski jezici.

b) TTCN je jezik za specifikaciju zahtjeva.

c) Semantika programa predstavlja specifikaciju procesa testiranja programske potpore.

d) UML je jezik za modeliranje.

e) P i Q predstavljaju instrukcije u uređenoj trojci {P}C{Q} kod Floyd-Hoare logike.

A. 1
B. 2
C. 3
D. 4

Zadatak 2. Osnovno načelo rada XSLT transformacije obuhvaća četiri faze. Navedite ispravan redoslijed sljedećih faza transformacija:

1. kreira se prazno izlazno stablo
2. učitava se XSLT (popis predložaka)
3. učitava se ulazno stablo
4. odabire se korijenski čvor u ulaznom stablu i traži se predložak za njega

A. 4 – 2 – 1 – 3
B. 3 – 2 – 1 – 4
C. 1 – 3 – 4 – 2
D. 2 – 3 – 4 – 1

Zadatak 3. Bilješke razlikujemo po namjeni, trajanju i broju parametara. Što je od ponuđenih odgovora točno za navedenu bilješku?

```
@Copyright( "2004, Dave Landers" )
```

A. Ima jedan parametar.
B. Ima dva parametra.
C. Namjenjena je označavanju metode.
D. Upotrebljava se samo u izvornom kôdu.

Zadatak 4. Neka je zadana datoteka `racunala.xml`:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<racunala>
  <racunalo tip="LAPTOP">
    <proizvodac>Unixoid</proizvodac>
    <cpu>2.4</cpu>
    <ram>2</ram>
    <cijena>12000.00</cijena>
  </racunalo >
  <racunalo tip="LAPTOP">
    <proizvodac>Jabuka</proizvodac>
    <cpu>2.66</cpu>
    <ram>4</ram>
    <cijena>15000.00</cijena>
  </racunalo >
  <racunalo tip="STOLNO">
    <proizvodac>ABC tech</proizvodac>
    <cpu>3.0</cpu>
    <ram>4</ram>
    <cijena>20000.00</cijena>
   </racunalo >
</racunala>
```

Koji od navedenih XQueryjevih upita ispravno dohvaća elemente `racunalo` koji imaju vrijednost elementa `cpu` veće od 2.8?

A.
```
for $x in doc("racunala.xml")/racunala/racunalo
where $x/cpu>2.8
order by $x/proizvodac
return $x/proizvodac
```

B.
```
for $x in doc("racunala.xml")/racunala/racunalo/*
where $x/cpu>2.8
order by $x/cpu
return $x/cpu
```

C.
```
for $x in doc("racunala.xml")/racunala/racunalo
where $x/cpu>2.8
order by $x/proizvodac
return $x
```

D.
```
for $x in doc("racunala.xml")/racunala/racunalo/*
where $x/racunalo>2.8
order by $x/cpu
return $x
```

Zadatak 5. Što od navedenog opisuje metodu **`assertEquals`**`(java.lang.String message, java.lang.Object expected, java.lang.Object actual)` ?

A. Vraća podatak tipa `true` ili `false`.

B. Uspoređuje navedena tri argumenta metode.

C. Vraća rezultat usporedbe objekata `expected` i `actual`.

D. Ne vraća ništa (`void`).

Zadatak 6. Što će se ispisati izvršavanjem sljedećeg programskog odsječka?

```
public static void main( String[] args ) {
      DifferentialCounter dc1 = new DifferentialCounter();
      XStream xs = new XStream();
      xs.alias("diff-counter", DifferentialCounter.class);
      String xml = xs.toXML(dc1);
      System.out.println(xml);
      DifferentialCounter dc2 = (DifferentialCounter) xs.fromXML(xml);
}
```

Klasa `DifferentialCounter` zadana je ovako:

```
package hr.fer.tel.ilj;

public class DifferentialCounter {
      private int cntr1;
      private int cntr2;

      public DifferentialCounter () {
            this.cntr2 = 1;
            this.cntr1 = 2;
      }
      …
}
```

A.

```
<diff-counter>
 <cntr1>2</cntr1>
 <cntr2>1</cntr2>
</diff-counter>
```

B.

```
<diff-counter>
  <cntr1>
        <counterState>2</counterState>
  </cntr1>
  <cntr2>
        <counterState>1</counterState>
  </cntr2>
</diff-counter>
```

C.

```
<hr.fer.tel.ilj.DifferentialCounter>
  <cntr1>
        <counterState>2</counterState>
  </cntr1>
  <cntr2>
        <counterState>1</counterState>
  </cntr2>
</hr.fer.tel.ilj.DifferentialCounter>
```

D.

```
<hr.fer.tel.ilj.DifferentialCounter>
  <cntr1>2</cntr1>
  <cntr2>1</cntr2>
</hr.fer.tel.ilj.DifferentialCounter>
```

Zadatak 7. Objasnite sljedeći izraz koji koristi alat EasyMock pri izradi lažnih (engl. *mock*) objekata:

```
EasyMock.expect (mock.helperMethod ("bok! kako je?"))
                .andReturn ("ma ne moze bolje.");
```

Zadatak 8. Odredite točan redoslijed dijelova životnog ciklusa EasyMockovog lažnog objekta:

1. CREATE
2. VERIFY
3. REPLAY
4. EXPECT

A. 4, 3, 1, 2
B. 1, 2, 4, 3
C. 1, 4, 3, 2
D. 4, 1, 2, 3

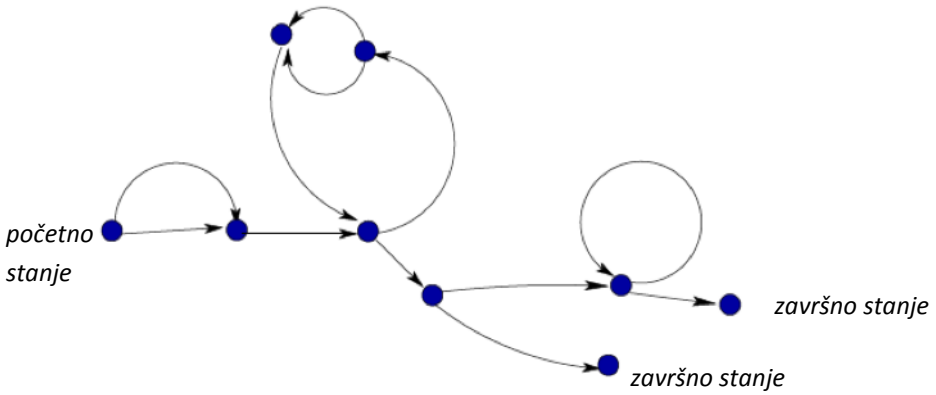
Zadatak 9. Neka je zadano sučelje:

```
public interface telephoneSwitch {
      void setInitialState(double state);
      double getCurrentState();
      void setCaller(String caller);
      String getCaller();
}
```

Koja od sljedećih metoda može pripadati implementaciji lažnog (*mock*) objekta za navedeno sučelje?

A. `EasyMock.expect(mock.setInitialSate(0)).andReturn(0);`
B. `EasyMock.expect(mock.getCurrentSate(0)).andReturn(0);`
C. `EasyMock.expect(mock.setCaller("Pero Perić")).andReturn("Pero Perić");`
D. Sve navedeno može pripadati implementaciji.

Zadatak 10. Koliko je minimalno putova potrebno testirati kako bi se ispitala svaka grana i svaki čvor programa opisanog sljedećim grafom stanja s označenim početnim i završnim stanjima programa?



A. 1
B. 2
C. 9
D. 12

Zadatak 11. Ukratko objasnite što radi sljedeći test napisan u TTCN-u ako pretpostavimo da je zadana struktura poruke `m_dnsQuestion` i `m_dnsQuestion` te definiran port `serverPort` testirajuće komponente `DnsClient`.

```
testcase TC_resolveEtsiWww() runs on DnsClient
{
   timer t_ack;

   serverPort.send(m_dnsQuestion("www.etsi.org"));
   t_ack.start(1.0);

   alt {
   [] serverPort.receive(mw_dnsAnswer("172.26.1.17")) {
      setverdict (pass);
   }
   [] serverPort.receive {
      setverdict(fail);
   }
   [] t_ack.timeout {
      setverdict(inconc);
   }
  }
  t_ack.stop;
}
```

Zadatak 12. Objasnite što je pogrešno u sljedećem programskom odsječku.

```
// Datoteka: Tocka.java
class Tocka{
     double x;  // x - koordinata točke u ravnini
     double y;  // y - koordinata točke u ravnini

     public Tocka(double a, double b){
           x=a;
           y=b;
      }
  }
// Datoteka: Test.java
public class Test{
    public static void main(String[] args){
       Tocka toc = new Tocka();
     }
}
```