Rešenja prvog kolokvijuma iz Operativnih sistema 2
Oktobar 2019.

1. (10 poena)

Trenutak	Izvršava se proces
15	B
25	C
35	D
55	D
65	C
75	D
95	B
110	D

2. (10 poena)

```
monitor DiningPhilosophers;
export startEating, stopEating;

  var
    forks : array 0..4 of boolean;
    canEat : array 0..4 of condition;

procedure startEating (i : integer);
begin
  var left, right : integer;
  left := i; right := (i+1) mod 5;
  if forks[left] then canEat[left].wait;
  forks[left] := true;
  if forks[right] then canEat[right].wait;
  forks[right] := true;
end;

procedure stopEating (i : integer);
begin
  var left, right : integer;
  left := i; right := (i+1) mod 5;
  forks[left] := false;
  forks[right] := false;
  canEat[left].signal;
  canEat[right].signal;
end;

begin
  var i: integer;
  for i:=0 to 4 do forks[i] := false;
end;
```

3. (10 poena)

```
public class Server {
    private final ServerSocket socket;
    private final DataAccess dataAccess;
    private Socket clientOwner = null;

    public Server() throws IOException {
        socket = new ServerSocket(5555);
        dataAccess = new DataAccess();
    }

    public void work() throws IOException {
        while(true) {
            Socket client = socket.accept();

            new RequestHandler(client, this).start();
        }
    }

    public synchronized void lock(Socket client) {
        while (clientOwner != null) {
            try {
                wait();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        clientOwner = client;
    }

    public synchronized void unlock(Socket client) {
        if (client == clientOwner) {
            clientOwner = null;
            notifyAll();
        }
    }

    public synchronized int[] read(Socket client, int address, int size) {
        while (client != clientOwner && clientOwner != null) {
            try {
                wait();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        return dataAccess.read(address, size);
    }

    public synchronized void write(Socket client, int address, int size, int[]
values) {
        while (client != clientOwner && clientOwner != null) {
            try {
                wait();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        dataAccess.write(address, size, values);
    }
}

public class RequestHandler extends Thread {
    private final Socket client;
    private final Server server;

    public RequestHandler(Socket client, Server server) {
        this.client = client;
```

```
        this.server = server;
    }

    public void run() {
        Service service = new Service(client);
        while(true) {
            // #<method>#arg0#...#[0]#[1]#...#[n - 1]#
            String[] args = service.receiveMessage().split("#");
            if (args[0].equals("lock")) {
                lock(args);
                service.sendMessage("OK");
            } else if (args[0].equals("lock")) {
                unlock(args);
                service.sendMessage("OK");
            } else if (args[0].equals("lock")) {
                write(args);
                service.sendMessage("OK");
            } else if (args[0].equals("lock")) {
                service.sendMessage(read(args));
            }
        }
    }

    public void lock(String[] args) {
        server.lock(client);
    }

    public synchronized void unlock(String[] args) {
        server.unlock(client);
    }

    public String read(String[] args) {
        int address = Integer.parseInt(args[1]);
        int size = Integer.parseInt(args[2]);
        return arrayToString(server.read(client, address, size));
    }

    private String arrayToString(int[] array) {
        StringBuilder sb = new StringBuilder();
        for (int x : array) {
            sb.append('#');
            sb.append(x);
        }
        sb.append('#');
        return sb.toString();

    }

    public void write(String[] args) {
        int address = Integer.parseInt(args[1]);
        int size = Integer.parseInt(args[2]);
        int[] values = new int[args.length - 3];
        for (int i = 4; i < args.length; i++) {
            values[i - 4] = Integer.parseInt(args[i]);
        }
        server.write(client,address, size, values);
    }
}
```

Klasa `Service` (`Usluga`) je data na vežbama.