Rešenja prvog kolokvijuma iz Operativnih sistema 2
Oktobar 2013.

1. (10 poena)

```
const int timesliceHP = ..., timesliceMP = ..., timesliceLP = ...;
enum Queue {HP,MP,LP};
extern PCB* const idle;

class Scheduler {
public:
  Scheduler ();
  PCB* get ();
  void put (PCB*, int wasBlocked);
private:
  PCB* head[3];  // 0-HP, 1-MP, 2-LP
  PCB* tail[3];
  static const int timeslice[3];
};

const int Scheduler::timeslice[] = {timesliceHP,timesliceMP,timesliceLP};

Scheduler::Scheduler () {
  for (int i=0; i<2; i++)
    head[i]=tail[i]=0;
}

PCB* Scheduler::get () {
  for (int i=0; i<2; i++)
    if (head[i]) {
      PCB* ret = head[i];
      head[i] = head[i]->next;
      if (head[i]==0) tail[i]=0;
      ret->next = 0;
      ret->timeslice=timeslice[i];
      ret->lastQueue=(Queue)i;
      return ret;
    }
  return idle;
}

void Scheduler::put (PCB* pcb, int wasBlocked) {
  if (pcb==0) return; // Exception!
  int i=0;
  if (!wasBlocked) {
    if (pcb->lastQueue==HQ) i=1;
    else i=2;
  }
  pcb->next = 0;
  if (tail[i]==0)
    tail[i] = head[i] = pcb;
  else
    tail[i] = tail[i]->next = pcb;
}
```

2. (10 poena)

```
monitor Mutex;
  export lock, unlock;

  var isLocked : boolean,
      isFree : condition;

  procedure wait ()
  begin
    if isLocked then isFree.wait();
    isLocked:=true;
  end;

  procedure unlock ()
  begin
    isLocked:=false;
    isFree.signal();
  end;

begin
  isLocked:=false;
end;
```

3. (10 poena)

Na serverskoj strani u klasi `Server` treba dodati sledeće atribute:

```
 public static boolean kraj = false;
 protected TaskExecutor executor;

public Server(int port, TaskExecutor executor) {
 this.executor = executor;
 ...

//poziv konstruktora new RequestHandler(clientSocket, executor);

public static void main(String args[]) {
 TaskExecutor executor =  new TaskExecutor();
 Server s = new Server(6001,executor);
 s.start();
 executor.start();
 ...
```

`RequestHandler` treba izmeniti na sledeći način:

```
public class RequestHandler extends Thread {
 ...
 protected TaskExecutor executor;

 ...
public RequestHandler(Socket clientSocket, TaskExecutor executor) {
  this.sock = clientSocket;
  this.executor = executor;
  ...
}

protected void processRequest(String request) {
  StringTokenizer st = new StringTokenizer(request, "#");
  int taskId = Integer.parseInt(st.nextToken());
  int priority = Integer.parseInt(st.nextToken());
  executor.add(new WorkerThread(taskId,priority,out));
}

public class TaskExecutor extends Thread {
 protected ArrayList<WorkerThread> list = new ArrayList<WorkerThread>();
```

```
 private synchronized WorkerThread removeMaxPriAndDoAging() {
  WorkerThread maxPriWT = null;
  while (list.isEmpty()) try { wait(); } catch (InterruptedException e) {}

  for (WorkerThread wt : list) {
   if(maxPriWT == null)
    maxPriWT = wt;
   else if (maxPriWT.priority < wt.priority) {
    if (maxPriWT.priority < WorkerThread.maxPri) maxPriWT.priority++;
    maxPriWT = wt;
   }
   else if (wt.priority < WorkerThread.maxPri) {
    wt.priority++;
   }
  }
  list.remove(maxPriWT);
  return maxPriWT;
 }

 public synchronized void add(WorkerThread wt) {
  list.add(wt);
  notifyAll();
 }

 public void run() {
  while (!Server.kraj) {
   WorkerThread maxPriWT = removeMaxPriAndDoAging();
   maxPriWT.start();
   maxPriWT.waitToFinish();
   maxPriWT.outStream.println("" + maxPriWT.taskId);
  }
 }
}
```