NITI

```
public class ThreadsTest {
      public static void main(String args[]) {
            BytePrinter bp1 = new BytePrinter();
            BytePrinter bp2 = new BytePrinter();
            BytePrinter bp3 = new BytePrinter();
            bp1.start();
            bp2.start();
            bp3.start();
            }
}
class BytePrinter extends Thread {
  public void run() {
    System.out.println("pocinjem!");
    for (int b = -10; b < 10; b++) {
     System.out.println(b);
    }
    System.out.println("gotovo!");
 }
}
```

```
public class NamedThreadsTest {
      public static void main(String[] args) {
      NamedBytePrinter frank = new NamedBytePrinter("Frank");
      NamedBytePrinter mary = new NamedBytePrinter("Mary");
      NamedBytePrinter chris = new NamedBytePrinter("Chris");

      frank.start();
      mary.start();
      chris.start();
      }
}
class NamedBytePrinter extends Thread {
  public NamedBytePrinter(String name) {
    super(name);
  }
  public void run() {
    System.out.println(this.getName() + ": pocinjem!");
    for (int b = -10; b < 10; b++) {
      System.out.println(this.getName() + ": " + b);
    }
    System.out.println(this.getName() + ": gotovo!");
  }
}
```

NITI I PRIORITET

```
public class SelfishRunner extends Thread {
    private int tick = 1;
    private int num;
    public SelfishRunner(int num) {
        this.num = num;
    }
    public void run() {
        while (tick < 400000) {
            tick++;
            if ((tick % 50000) == 0)
          System.out.println("Thread #" + num + ", tick = " + tick);
        }}}
public class RaceTest {
    private final static int NUMRUNNERS = 2;
    public static void main(String[] args) {
        SelfishRunner[] runners = new SelfishRunner[NUMRUNNERS];
        for (int i = 0; i < NUMRUNNERS; i++) {
            runners[i] = new SelfishRunner(i);
            runners[i].setPriority(2);
```

```
        }
        for (int i = 0; i < NUMRUNNERS; i++)
            runners[i].start();
    }}
```

```
public class PoliteRunner extends Thread {

    private int tick = 1;
    private int num;

    public PoliteRunner(int num) {
        this.num = num;
    }

    public void run() {
        while (tick < 400000) {
            tick++;
            if ((tick % 50000) == 0) {
                System.out.println("Thread #" + num + ", tick = " +
tick);
                yield();
            }
        }
    }}
public class RaceTest2 {

    private final static int NUMRUNNERS = 2;

    public static void main(String[] args) {
        PoliteRunner[] runners = new PoliteRunner[NUMRUNNERS];

        for (int i = 0; i < NUMRUNNERS; i++) {
            runners[i] = new PoliteRunner(i);
            runners[i].setPriority(2);
        }
        for (int i = 0; i < NUMRUNNERS; i++)
            runners[i].start();
    }}
```

APLETI

```
import java.applet.*;
import java.awt.*;

public class HelloWorld extends Applet {
  Label helloLabel=new Label("Zdravo !");
    public void init() {
       setBackground(Color.yellow);
       add(helloLabel);
    }
}
<HTML>
<HEAD>
<TITLE>Hello to Everyone!</TITLE>
</HEAD>
<BODY>
<P>My Java Applet says:
<APPLET CODE="HelloApplet.class" WIDTH=150 HEIGHT=25>
</BODY>
</HTML>
```

```
import java.awt.event.MouseListener;
import java.awt.event.MouseEvent;
import java.applet.Applet;
import java.awt.Graphics;

public class SimpleClick extends Applet implements MouseListener {
    StringBuffer buffer;

    public void init() {
      addMouseListener(this);
      buffer = new StringBuffer();
        addItem("initializing... ");
    }
    public void start() {
        addItem("starting... ");
    }
    public void stop() {
        addItem("stopping... ");
    }
    public void destroy() {
        addItem("preparing for unloading...");
    }
    void addItem(String newWord) {
        System.out.println(newWord);
        buffer.append(newWord);
        repaint();
    }
    public void paint(Graphics g) {
        g.drawRect(0, 0, getSize().width - 1, getSize().height - 1);
        g.drawString(buffer.toString(), 5, 15);
    }
    public void mouseEntered(MouseEvent event) {}
    public void mouseExited(MouseEvent event) {}
    public void mousePressed(MouseEvent event) {}
    public void mouseReleased(MouseEvent event) {}
    public void mouseClicked(MouseEvent event) {
      addItem("click!... ");
    }
}
```

APLETI I NITI

```
import java.awt.Graphics;
import java.util.*;
import java.text.DateFormat;
import java.applet.Applet;

public class Clock extends Applet implements Runnable {
    private Thread clockThread = null;
    public void start() {
        if (clockThread == null) {
            clockThread = new Thread(this, "Clock");
            clockThread.start();
        }
    }
    public void run() {
        Thread myThread = Thread.currentThread()
        while (clockThread == myThread) {
            repaint();
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e){
            }
        }
    }
    public void paint(Graphics g) {
        // get the time and convert it to a date
        Calendar cal = Calendar.getInstance();
```

```
            Date date = cal.getTime();
            // format it and display it
            DateFormat dateFormatter = DateFormat.getTimeInstance();
            g.drawString(dateFormatter.format(date), 5, 10);
        }
        // overrides Applet's stop method, not Thread's
        public void stop() {
            clockThread = null;
        }
    }
```

SINHRONIZACIJA PRIMER IZ SKRIPTE

```
class ProizvodjacKorisnikTest {
   public static void main(String args[]) {
      Skladiste skladiste = new Skladiste();
      Proizvodjac p1 = new Proizvodjac(skladiste, 1);
      Korisnik k1 = new Korisnik(skladiste,1 ) ;
      p1.start() ;
      k1.start() ;   }}

class Skladiste {
private int sadrzaj;
private boolean otkljucano = false;
public synchronized int uzmi() {
   while (otkljucano == false) {
     try {
       wait();
     } catch (InterruptedException e) { }
   }
       otkljucano = false;
      return sadrzaj;
  }
public synchronized void stavi(int vrednost) {
  sadrzaj = vrednost;
  otkljucano = true;
  notify();
}
}
class Korisnik extends Thread {
 private Skladiste mojeskladiste;
 private int broj;

 public Korisnik(Skladiste sk, int broj) {
    mojeskladiste = sk;
    this.broj = broj;
 }
 public void run() {
    int procitani_broj = 0;
    for (int i = 0; i < 10; i++) {
      procitani_broj = mojeskladiste.uzmi ();
      System.out.println("Korisnik#  "
        + this.broj + " je uzeo  : "
        + procitani_broj);
    }
  }
}
class Proizvodjac extends Thread {
   private Skladiste mojeskladiste;
   private int broj;
   public Proizvodjac(Skladiste sk, int broj) {
      mojeskladiste = sk;
      this.broj = broj;
   }
   public void run() {
      for (int i = 0; i < 10; i++) {
         mojeskladiste.stavi(i);
         System.out.println("Proizvodjac #"
            + this.broj + " je stavio: " + i);
         try {
            sleep((int)(Math.random() * 100));
         } catch (InterruptedException e) { }
      }
 }
```