Struktura e të dhënave

Shablonet e përsëritjes: Iterimi dhe rekursioni

Faton M. Berisha

fberisha@uni-pr.edu

Universiteti i Prishtinës

Fakulteti i Shkencave Matematike–natyrore

Referencat

Web sajte:

- `http://fberisha.netfirms.com`
- `www.cis.ksu.edu/santos/schmidt/ppj`
- `java.sun.com`

Literatura:

- D. Schmidt, *Programming principles in Java: Archtecture and Interfaces*.
- T. Budd, *Classic data structures in Java*, Addison Wesley, 2001.
- I. Horton, *Beginning Java 2*, Wrox Press, 2000.

Shablonet e përsëritjes: Iterimi dhe rekursioni

Objektivat:

- Të studiohen teknikat dhe zbatimet e përsëritjes në programim.
- Të studiohen urdhërat `while` dhe `for` për shkruarjen e shablloneve standarde dhe klasike të përsëritjes, së quajtur iteracion.
- Të tregohet se si të zbatohet përsëritja në aplikacione të cilat shfrytëzojnë disenj të orientuar nga objekte.
- Të studiohet një formë tjetër e përsëritjes, invokimi rekursiv i metodës, si teknikë për zgjidhjen e problemeve duke zgjidhur në mënyrë të përsëritur nënprobleme më të thjeshta.

Urdhëri `while`

Përsëritje: kryerje e përsëritur e një aksioni.

Një strukturë kontrolluese për përsëritje është *urdhëri* `while`.

Sintaksa e urdhërit `while`:

`while (`*`TESTI`*`) { `*`TRUPI`*` }`
ku *`TESTI`* është shprehje e tipit `boolean`, kurse *`TRUPI`* varg urdhërash.

Semantika:

1. Llogaritet vlera e shprehjes *`TESTI`*.
2. Në qoftë se vlera e *`TESTI`* është `true`, atëherë ekzekutohet *`TRUPI`* dhe procesi përsëritet duke rifilluar me hapin 1.
3. Në qoftë se vlera e *`TESTI`* është `false`, atëherë shpërfillet *`TRUPI`*, dhe përsëritja përfundon.

Urdhëri `while` – *Vazhdim*

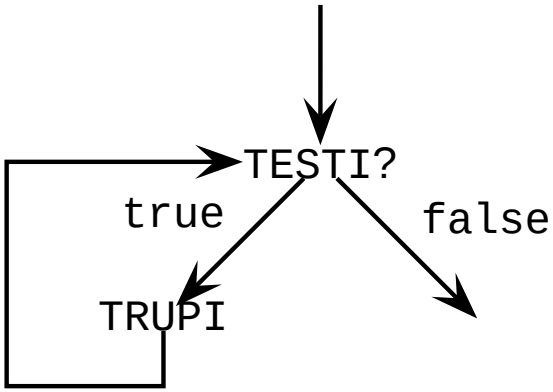


Figura 1. Skema e rrjedhës së kontrollit të përcaktuar nga një *while-cikël*

Urdhëri `while` *– Vazhdim*

```
public static void main(String[] args)
{ int i = 1;
  while ( i <= 20 )
  { System.out.println("1/" + i + " = " + 1.0 / i);
    i = i + 1;
  }
}
```

Urdhëri `while` *– Vazhdim*

Iterim: llogaritje e përsëritur e kryer nga një cikël; gjithashtu i referohet edhe një ekzekutimi të trupit të një cikli.

Iterim definitiv: iterimi ku numri i iterimeve të ciklit dihet në qastin kur cikli fillohet.

Iterim indefinitiv: iterim i cili nuk është definitiv.

Divergjencë: iterim i pafundmë.

Iterimi definitiv

Një shembull standard: Të konstruktohet një metodë për llogaritjen e shumës

$$sum(n) = 0 + 1 + 2 + \ldots + n.$$

Algoritmi:

1. Inicializo `total = 0` dhe një variabël të re `i = 0`.
2. Përderisa `i` nuk është e barabartë me `n`, vepro si vijon:
   - Rrite `i` për 1.
   - Shto `i` në `total`.
   - Përsërit hapin 2.
3. Kthe vlerën e `total`.

Iterimi definitiv – Vazhdim

```
/** Llogarit shumën 0+1+2+...+n.
 *  @param n - kufiri i sipërm; jonegativ
 *  @return shuma */
public int sum(int n)
{ int total = 0;
  int i = 0;
  while ( i != n)
  { // invariantë: total == 0+1+...+i
    i = i + 1;
    total = total + i;
  }
  return total;
}
```

Iterimi definitiv – Vazhdim

Veti invariante (invariantë): fakt logjik i cili është i saktë në fillimin dhe në mbarimin e secilit iteracion të ciklit; shfrytëzohet për dokumentim të ciklit dhe për vërtetim të korrektësisë së ciklit.

Numërues cikli (variabël kontrolluese): variabël vlera e së cilës memoron numrin e iterimeve të ciklit; tipikisht shfrytëzohet në testin e ciklit për të përfunduar iterimin.

Iterimi definitiv – Vazhdim

```
int n == 4   int total == 0   int i == 0
> while ( i != n )
{ i = i + 1;
  total = total + i;
}
```

```
int n == 4   int total == 0   int i == 0
while ( true )
{ > i = i + 1;
  total = total + i;
}
```

```
int n == 4   int total == 1   int i == 1
while ( true )
{ ...
>}
```

Iterimi definitiv – Vazhdim

```
int n == 4   int total == 1   int i == 1
> while ( i != n )
{ i = i + 1;
  total = total + i;
}
```

```
int n == 4   int total == 1   int i == 1
while ( true )
{ > i = i + 1;
  total = total + i;
}
```

```
int n == 4   int total == 10   int i == 4
> while ( i != n )
{ i = i + 1;
  total = total + i;
}
```

Iterimi definitiv – Vazhdim

```
int n == 4   int total == 10   int i == 4
while ( false )
{ ... }
> return total;
```

Iterimi definitiv – Vazhdim

Shablloni i iterimit definitiv më së shpeshti ka formëm

```
int i = VLERA_INICIALE;
while ( TESTI_SIPAS_i )
{ ITERIMI;
  RRITE_i;
}
```

Ndonjëherë variabla kontrolluese inkrementohet në fillim të iteracionit:

```
int i = VLERA_INICIALE;
while ( TESTI_SIPAS_i )
{ RRITE_i;
  ITERIMI;
}
```

Divergjenca

Për shembull, invokimi i metodës `sum` me parametër aktual negativ, shkakton divergjencë (p.sh., `sum(-1)`).

Pothuajse gjithmonë, divergjenca është një sjellje e padëshiruar.

Divergjenca – Vazhdim

```
/** Llogarit shumën 0+1+2+...+n.
 *  @param n - kufiri i sipërm; jonegativ
 *  @return shuma për n jonegativ; -1 për n negativ */
public int sum(int n)
{ int total = 0;
  if ( n < 0 )
  { System.out.println("Gabim në shumim: int negativ " + n);
    total = -1;
  }
  else
  { int i = 0;
    while ( i != n)
    { // ... sikur më parë
    }
  }
  return total;
}
```

Iterimi indefinitiv: procesimi i hyrjes

Shablloni i procesimit të hyrjes:

```
boolean processing = true;
while ( processing )
{ LEXO_NJË_INPUT_TRANSAKCION;
  if ( INPUT_INDIKON_NDALESË )
  { processing = false; }
  else { PROCESO_TRANSAKCIONIN; }
}
```

Iterimi indefinitiv: procesimi i hyrjes – Vazhdim

```
/** Llogarit vlerat reciproke.
 *  Input: një varg numrash të plotë jonegativë */
import javax.swing.*;
public class PrintReciprocals
{ public static void main(String[] args)
  { boolean processing = true;
    while ( processing )
    { // deri tani të gjitha vlerat reciproke korrekte
      int input = new Integer(JOptionPane.showInputDialog
        ("Jepni një numër të plotë jonegativ:")).intValue();
      if ( input <= 0 )
      { processing = false; }
      else { JOptionPane.showMessageDialog(null,
                "1/" + input + " = " + 1.0/input);
           }
    }
  }
}
```

Iterimi indefinitiv: kërkimi

Algoritmi për kërkimin e stringut `s` për karakterin `c`:

1. Fillo me skajin e majtë të stringut, d.m.th., vëj `index = 0`.
2. Përderisa `c` nuk është gjetur ende dhe ka akoma karakterë të paekzaminuar:
   (a) Në qoftë se `s.charAt(index) == c`, atëherë karakteri është gjetur dhe kërkimi mund të përfundohet.
   (b) Përndryshe rrit `index` dhe përsërit hapin 2.

Iterimi indefinitiv: kërkimi – Vazhdim

```
/** Alokon paraqitjen e parë nga e majta të karakterit në string.
 *  @param c - karakteri për t'u gjetur
 *  @param s - stringu për t'u kërkuar
 *  @return indeksi i paraqitjes;
 *    -1 në qoftë se c nuk paraqitet në s */
public int findChar(char c, String s)
{ boolean found = false;
  int index = 0;
  while ( !found && index < s.length() )
  { // për found==false c nuk është në pozitat 0...(index-1)
    // për found==true c është s.charAt(index)
    if (s.charAt(index) == c)
    { found = true; }
    else { index = index + 1; }
  }
  if ( !found ) { index = -1; }
  return index;
}
```

Iterimi indefinitiv: kërkimi – Vazhdim

Shablloni i kërkimit të një koleksioni:

```
boolean found = false;
PËRCAKTO_ELEMENTIN_E_PARË;
while ( !found && KA_ENDE_ELEMENTE )
{ EKZAMINO_ELEMENTIN;
  if ( ELEMENTI_I_KËRKUAR )
  { found = true; }
  else { PËRCAKTO_ELEMENTIN_VIJUES; }
}
```

Iterimi indefinitiv: kërkimi – Vazhdim

```
/** Kontrollon një numër të plotë se a është i thjesht.
 *  @param n - numri i plotë > 1
 *  @return pjesëtuesin më të madh jotrivial të numrit n;
 *    -1 nëse argumenti është jovalid */
public int isPrime(int n)
{ int answer = -1;
  if ( n < 2 )
  { System.out.println("Gabim: Argument jovalid " + n);}
  else
```

Iterimi indefinitiv: kërkimi – Vazhdim

```
  { boolean found = false;
    int current = n / 2;
    while ( !found && current > 1 )
    { // për found==false n/2, n/2-1,..., current+1
      //   nuk janë pjesëtues
      // për found == true current është pjesëtues i n
      if ( n % current == 0 )
      { found = true; }
      else { current = current - 1; }
    }
    if ( found )
    { answer = current; }
    else { answer = 1; }
  }
  return answer;
}
```

Përfundimi i ciklit me `break`

Urdhëri `break` terminon ciklin në pikën ku është arritur vendimi.

```
public int findChar(char c, String s)
{ int index = 0;
  while ( index < s.length() )
  { // c nuk është asnjë nga karakterët 0...(index-1)
    if (s.charAt(index) == c)
    { break; }
    else { index = index + 1; }
  }
  if ( !(index < s.length()) )
  { index = -1; }
  return index;
}
```

Urdhëri `for`

Sintaksa e *urdhërit* `for`*:*

```
for ( int i = VLERA_INICIALE; TESTI; RRITE_i)
{ TRUPI }
```

Semantika është sikur e shabllonit të iterimit definitiv:

```
{ int i = VLERA_INICIALE;
  while ( TESTI )
  { TRUPI;
    RRITE_i;
  }
}
```

përveç se skopi i `i` shtrihet vetëm përbrenda trupit të urdhërit `for`.

Urdhëri `for` *– Vazhdim*

Shembuj:

```
for ( int i = 1; i <= 20; i++ )
{ // invariantë: janë llogaritur vlerat reciproke të 1...(i-1)
  System.out.println("1/" + i + " = " + 1.0 / i); }

for ( int i = s.length() - 1; i >= 0; i-- )
{ // invariantë: janë shtypur karakterët te s.length()-1...(i+1)
  System.out.println(s.charAt(i)); }

for ( int i = 1; i <= n; i++ )
{ // invariantë: total == 0+1+...+(i-1)
  total += i; }
```

Urdhëri `for` *– Vazhdim*

```
for ( int i = 0; i <= 5; i++ )
{ // invariantë: janë shtypur 0*x...(i-1)*x për x nga 0...5
  for ( int j = 0; j <= 5; j++ )
    { // invariantë: janë shtypur i*0...i*(j-1)
      System.out.print(i + "*" + j + "=" + (i * j));
    }
  System.out.println();
}
```

Shkruarja dhe testimi i cikleve

Shembuj gabimesh semantike në cikle:

```
int i = 2;
while ( i != 0 ); // përsëritet urdhër bosh
{ i--; }

int total = 0; int i = 0;
while ( i <= n )
{ i++;
  total += i;
} // një iterim tepër

int total = 0; int i = 1;
while ( i <= n )
{ i++;
  total += i;
} // vlera iniciale
```

Shkruarja dhe testimi i cikleve – Vazhdim

```
int total = 0;
int i = 0;
while ( i != n )
{ total += i;
  i++;
} // një iterim mangu

double d = 0.0;
while ( d != 1.0 )  // krahasim në barazim i dy double
{ d = d + (1.0 / 13);
  System.out.println(d);
}
```

Testimi i cikleve duhet të përfshijë test vlera të cilat shkaktojnë që cikli të përfundojë me 0 iterime, 1 iterim, iterim të shumfishtë, si dhe të tilla që i bëjnë të divergjojnë.

OO disenji: Komponentet, kolaboratorët dhe përgjegjësitë

- Komponentet: klasët e nevojshme për të modeluar problemin (identifikimi i „aktorëve“, „karakterëve“).
- Përgjegjësitë: metodat që duhet t’i ketë klasa (identifikimi i „aftësive“ të aktorëve).
- Kolaboratorët: klasat të cilave iu dërgon mesazhe klasa (identifikimi i „interaksionit“ ndërmjet aktorëve).

CRC skedat

CRC (Components/Responsibilities/Colaborators) skedë:
skedë indeksi e cila përmban informata disenjimi mbi një klasë: emrin, përgjegjësitë dhe emrat e klasave kolaboratore.

Case study: Animimi i topit kërcyes

Animim: aplikacion i cili afishon objekte lëvizëse.

`MovingBall`	modelon një top lëvizës
Përgjegjësitë:	lëviz "një hap" brenda një njësie, duke ndryshuar drejtimin kur të preket ndonjë barierë
Kolaboratorët:	`Box`

Tabela 1. CRC kartela për topin lëvizës

`Box`	modelon një kuti katrore
Përgjegjësitë:	përgjegjet se a është pozita e dhënë në kontakt me ndonjërin nga muret

Tabela 2. CRC kartela për kutinë

Case study: Animimi i topit kërcyes – Vazhdim

`class MovingBall`	modelon një top lëvizës
Metodat:	
`move()`	lëviz "një hap" brenda një njësie kohore, varësisht nga shpejtësia e vetë, duke ndryshuar drejtimin kur të preket ndonjë mur
`int xPosition()`	kthen pozitën horizontale
`int yPosition()`	kthen pozitën vertikale
`int radiusOf()`	kthen rrezen e topit
Kolaboratorët:	`Box`

Tabela 3. Specifikimi i interfejsit për topin lëvizës

Case study: Animimi i topit kërcyes – Vazhdim

`class Box`	modelon një kuti katrore
Metodat:	
`boolean inHorizontalContact (int xPosition)`	kthen se a është koordinata horizontale në kontakt me njërin nga muret vertikale
`boolean inVerticalContact (int yPosition)`	kthen se a është koordinata vertikale në kontakt me njërin nga muret horizontale
`int sizeOf()`	kthen madhësinë e kutisë

Tabela 4. Specifikimi i interfejsit për kutinë

Case study: Animimi i topit kërcyes – Vazhdim

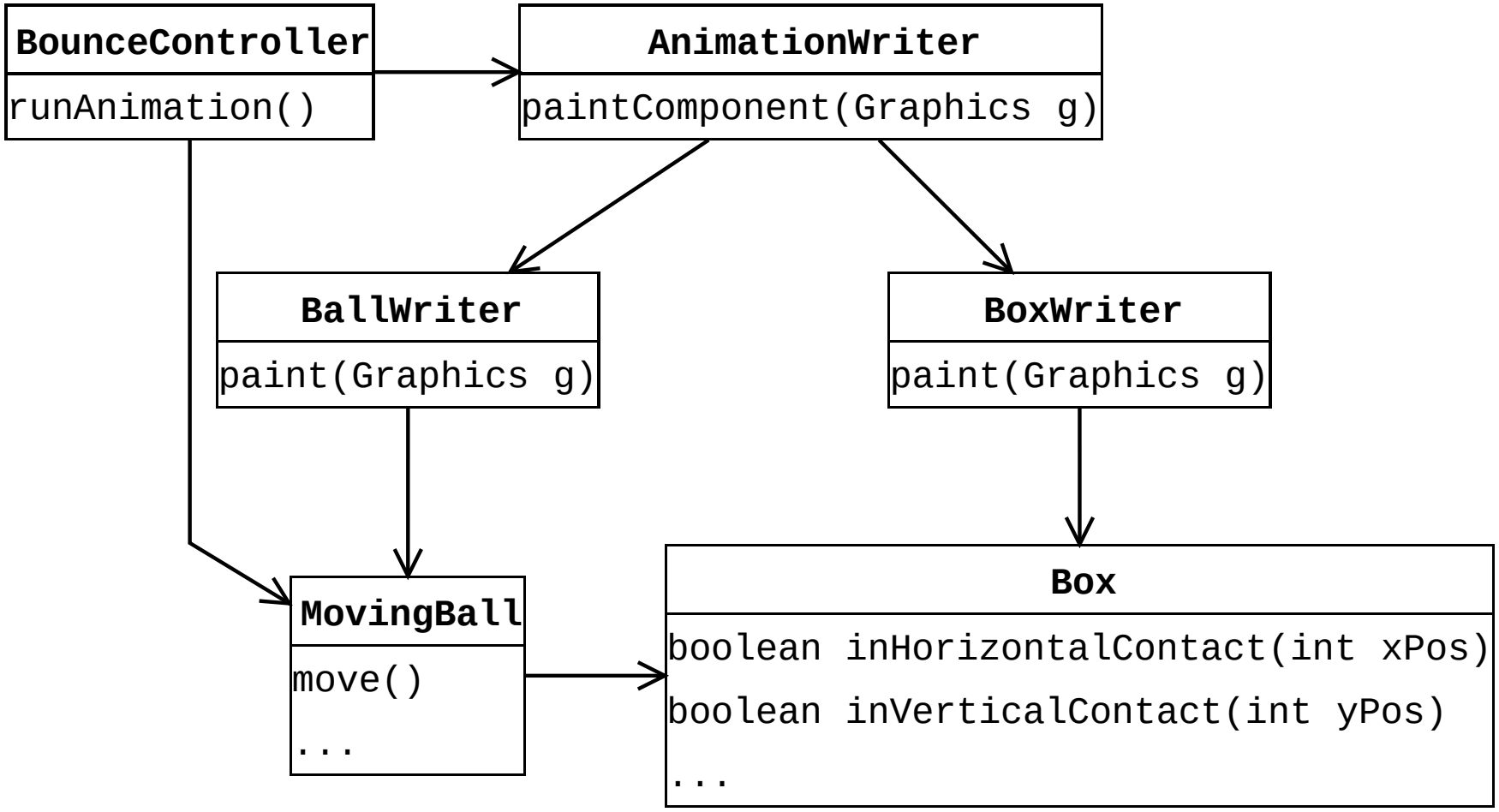


Figura 2. Arkitektura e animimit të topit kërcyes

Case study: Animimi i topit kërcyes – Vazhdim

```
/** Modelon kutinë në të cilën ndodhet topi. */
public class Box
{ private int boxSize;
  /** Ndërton kutinë.
   *  @param size - madhësia e kutisë */
  public Box(int size)
  { boxSize = size; }
  /** Kthen se a ka kontakt koordinata horizontale me murin.
   *  @param xPosition - pozita që ekzaminohet
   *  @return true në qoftë se ka kontakt;
   *    false përndryshe */
  public boolean inHorizontalContact(int xPosition)
  { return xPosition <= 0 || xPosition >= boxSize; }
```

Case study: Animimi i topit kërcyes – Vazhdim

```
  /** Kthen se a ka kontakt koordinata vertikale me murin.
   *  @param yPosition - pozita që ekzaminohet
   *  @return true në qoftë se ka kontakt;
   *    false përndryshe */
  public boolean inVerticalContact(int yPosition)
  { return yPosition <= 0 || yPosition >= boxSize; }
  /** Kthen madhësinë e kutisë. */
  public int sizeOf()
  { return boxSize; }
}
```

Case study: Animimi i topit kërcyes – Vazhdim

```
/** Modelon një top lëvizës. */
public class MovingBall
{ private int xPos;
  private int yPos;
  private int radius;
  private int xVelocity = 5;
  private int yVelocity = 2;
  private Box container;
```

Case study: Animimi i topit kërcyes – Vazhdim

```
/** Konstrukton topin.
 *  @param xInit - pozita horizontale e qendrës së topit
 *  @param yInit - pozita vertikale e qendrës së topit
 *  @param r - rrezja e topit
 *  @param box - kutia nëpër të cilën lëviz topi */
public MovingBall(int xInit, int yInit, int r, Box box)
{ xPos = xInit;
  yPos = yInit;
  radius = r;
  container = box;
}
/** Kthen pozitën horizontale të topit. */
public int xPosition()
{ return xPos; }
```

Case study: Animimi i topit kërcyes – Vazhdim

```
  /** Kthen pozitën vertikale të topit. */
  public int yPosition()
  { return yPos; }
  /** Kthen rrezen e topit. */
  public int radiusOf()
  { return radius; }
  /** Lëviz topin një hap. */
  public void move()
  { xPos += xVelocity;
    if ( container.inHorizontalContact(xPos) )
    { xVelocity = -xVelocity; }
    yPos += yVelocity;
    if ( container.inVerticalContact(yPos) )
    { yVelocity = -yVelocity; }
  }
}
```

Case study: Animimi i topit kërcyes – Vazhdim

```
import javax.swing.*;
import java.awt.*;
/** Afishon një kuti me një top në të. */
public class AnimationWriter extends JPanel
{ private BoxWriter boxWriter;
  private BallWriter ballWriter;
```

Case study: Animimi i topit kërcyes – Vazhdim

```
  /** Konstrukton pamjen e kutisë dhe topit.
   * @param bx - shkruesi i kutisë
   * @param bl - shkruesi i topit
   * @param size - madhësia e kornizës */
  public AnimationWriter(BoxWriter bx, BallWriter bl, int size)
  { boxWriter = bx;
    ballWriter = bl;
    JFrame f = new JFrame();
    f.getContentPane().add(this);
    f.setTitle("Topi kërcyes");
    f.setSize(size, size);
    f.setVisible(true);
  }
  /** Vizaton kutinë dhe topin.
   * @param g - penda grafike */
  public void paintComponent(Graphics g)
  { boxWriter.paint(g); ballWriter.paint(g); }
}
```

Case study: Animimi i topit kërcyes – Vazhdim

```
import java.awt.*;
/** Afishon kutinë. */
public class BoxWriter
{ private Box box;
  /** Konstrukton shkruesin.
   * @param b - kutia që afishohet */
  public BoxWriter(Box b)
  { box = b; }
  /** Vizaton kutinë.
   * @param g - penda grafike */
  public void paint(Graphics g)
  { int size = box.sizeOf();
    g.setColor(Color.white);
    g.fillRect(0, 0, size, size);
    g.setColor(Color.black);
    g.drawRect(0, 0, size, size);
  }
}
```

Case study: Animimi i topit kërcyes – Vazhdim

```
import java.awt.*;
/** Afishon një top lëvizës. */
public class BallWriter
{ private MovingBall ball;
  private Color color;
  /** Konstrukton shkruesin.
   * @param b - topi që afishohet
   * @param c - ngjyra e topit */
  public BallWriter(MovingBall b, Color c)
  { ball = b;
    color = c;
  }
```

Case study: Animimi i topit kërcyes – Vazhdim

```
  /** Vizaton topin.
   * @param g - penda grafike */
  public void paint(Graphics g)
  { g.setColor(color);
    int radius = ball.radiusOf();
    g.fillOval(ball.xPosition() - radius,
      ball.yPosition() - radius, 2 * radius, 2 * radius);
  }
}
```

Case study: Animimi i topit kërcyes – Vazhdim

```
/** Kontrollon një top lëvizës brenda një kutie. */
public class BounceController
{ private MovingBall ball;
  private AnimationWriter writer;
  /** Inicializon kontrolluesin.
   * @param b - objekti model
   * @param w - objekti output view */
  public BounceController(MovingBall b, AnimationWriter w)
  { ball = b;
    writer = w;
  }
```

Case study: Animimi i topit kërcyes – Vazhdim

```
  /** Ekzekuton animimin përmes orës interne. */
  public void runAnimation()
  { while ( true )
    { delay(20);
      ball.move();
      writer.repaint();
    }
  }

  private void delay(int milisecs)
  { try { Thread.sleep(milisecs); }
    catch (InterruptedException e) { }
  }
}
```

Case study: Animimi i topit kërcyes – Vazhdim

```
import java.awt.*;
/** Konstrukton dhe starton objektet në një animim. */
public class TestBouncingBall
{ public static void main(String[] args)
  { int boxSize = 200;
    int ballRadius = 6;
    Box box = new Box(boxSize);
    MovingBall ball = new MovingBall(boxSize / 3, boxSize / 5,
      ballRadius, box);
    BallWriter ballWriter = new BallWriter(ball, Color.red);
    BoxWriter boxWriter = new BoxWriter(box);
    AnimationWriter writer = new AnimationWriter(boxWriter,
      ballWriter, boxSize);
    new BounceController(ball, writer).runAnimation();
  }
}
```

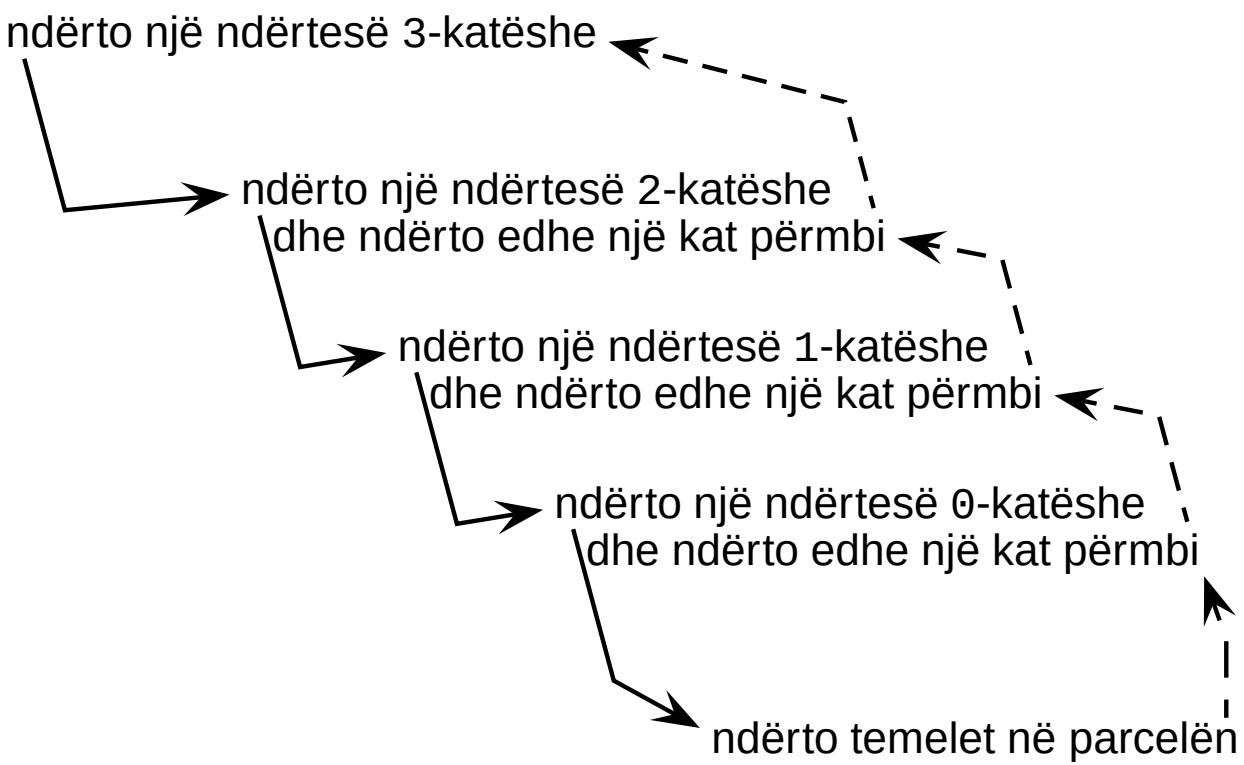
Rekursioni

Definim rekursiv: definim metode ashtu që të zbatojë invokim rekursiv, d.m.th., t'i dërgojë mesazhe vetvetës. Përdoret për të zgjidhur një problem ashtu që së pari të zgjidhen versione më të thjeshta të problemit, duke shfrytëzuar pastaj rezultatet për të zgjidhur problemin në tërësi.

Algoritmi për të ndërtuar një ndërtesë me sado kate:

- Për të ndërtuar një ndërtesë `(n+1)`-katëshe ndërto një ndërtesë n-katëshe dhe ndërto edhe një kat përmbi.
- Për të ndërtuar një ndërtesë `0`-katëshe ndërto temelet në parcelën.

Rekursioni – Vazhdim



Rekursioni – Vazhdim

```
/** Llogarit n!
 * @param n - numër i plotë nga 0..20
 * @return n! për n nga 0..20; -1 përndryshe */
public long factorial(int n)
{ long answer;
  if ( n < 0 || n > 20 )
  { answer = -1; }
  else if ( n == 0 )
       { answer = 1; }
       else { answer = n * factorial(n-1); }
  return answer;
}
```

Rekursioni – Vazhdim

```
factorial
{ int n == 3   long answer == ?
1> if ( n < 0 || n > 20 )
   { answer = -1; }
   else if ( n == 0 )
        { answer = 1; }
        else { answer = n * factorial(n-1); }
   return answer;
}
```

Rekursioni – Vazhdim

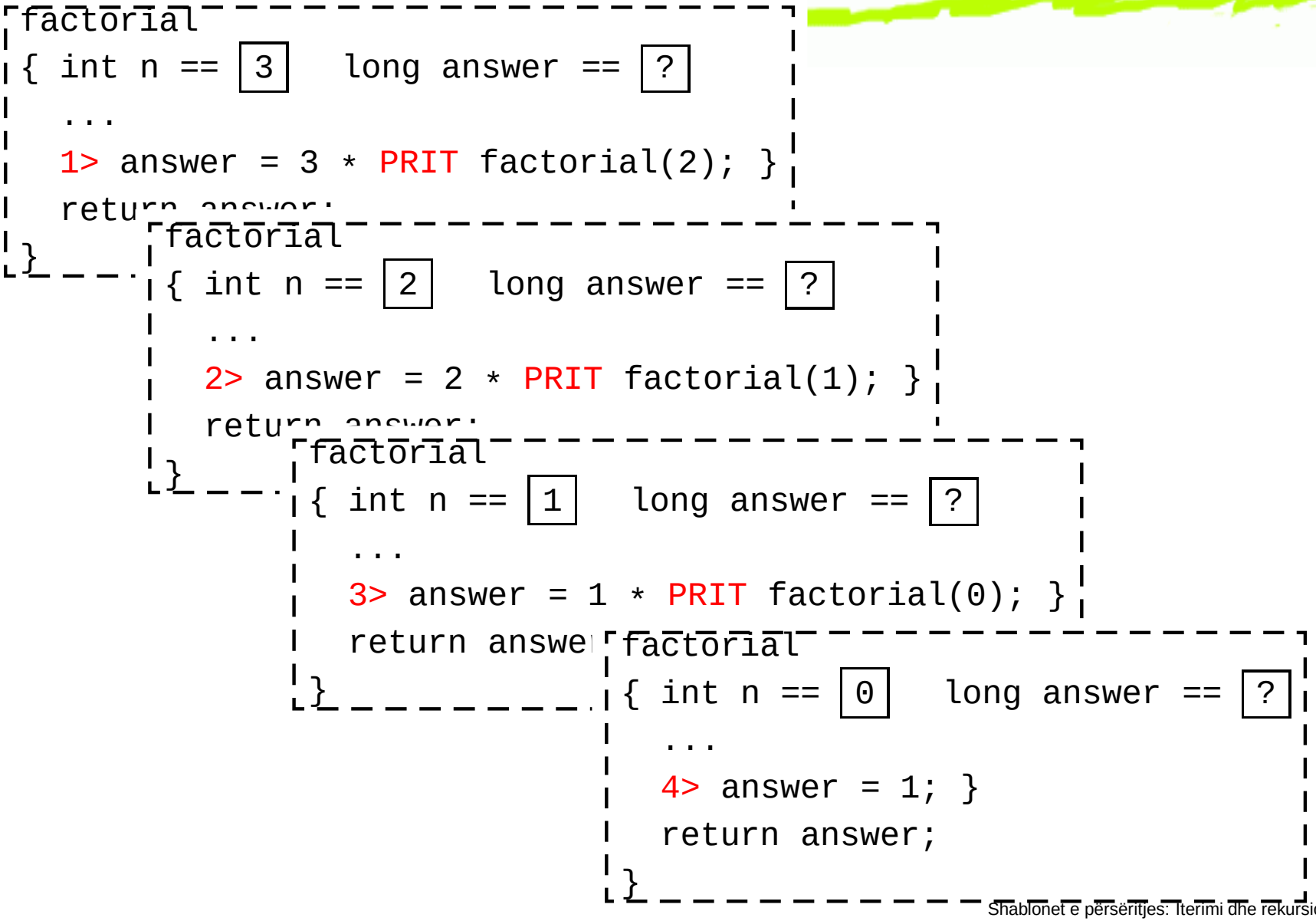
```
factorial
{ int n == 3   long answer == ?
  ...
  1>          answer = 3 * factorial(n-1); }
  return answer;
}
```

Rekursioni – Vazhdim

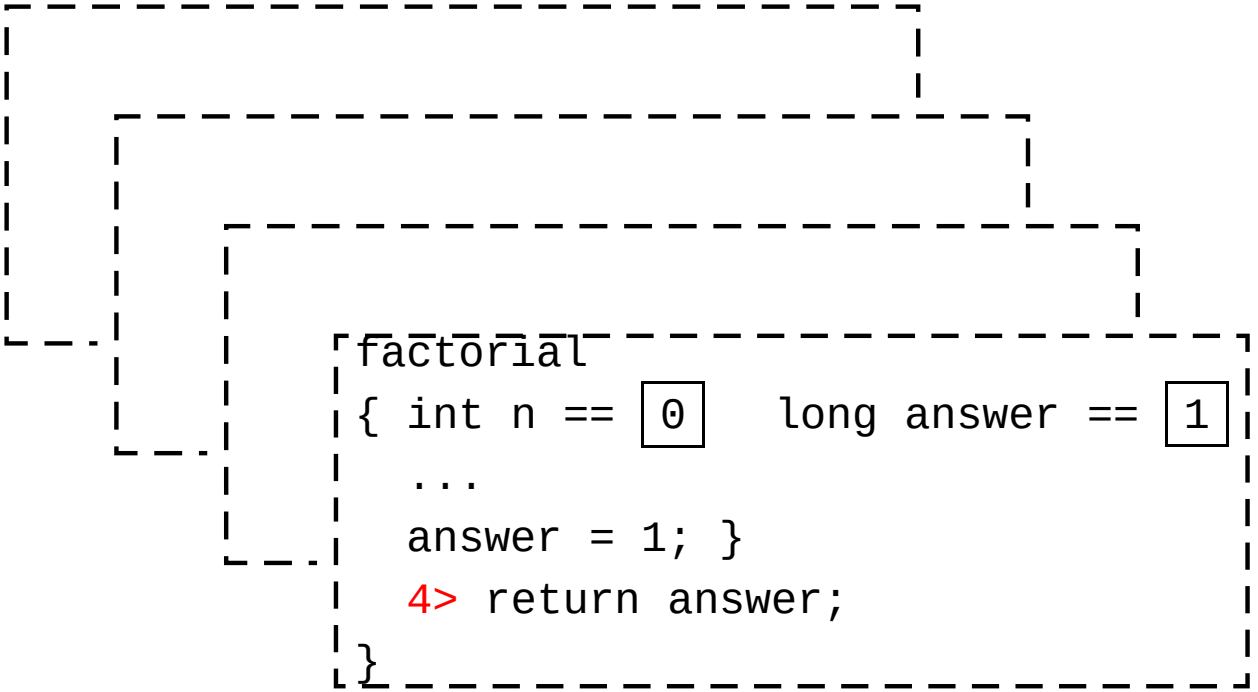
```
factorial
{ int n == 3    long answer == ?
  ...
  1> answer = 3 * PRIT factorial(2); }
  return answer;
}
```

```
factorial
{ int n == 2    long answer == ?
  2> if ( n < 0 || n > 20 )
  { answer = -1; }
  else if ( n == 0 )
       { answer = 1; }
       else { answer = n * factorial(n-1); }
  return answer;
}
```

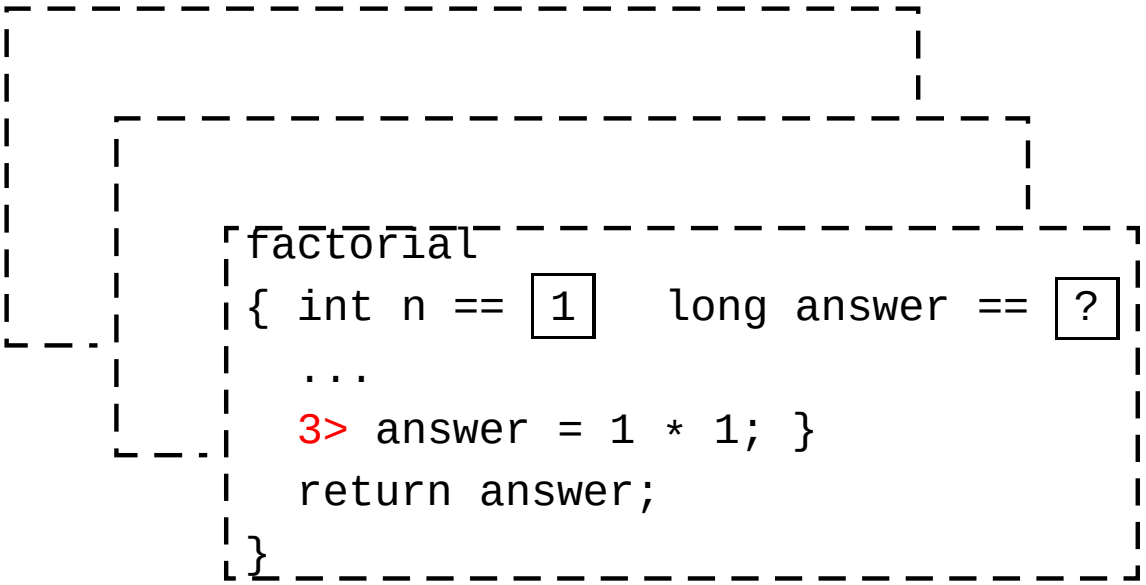
Rekursioni – Vazhdim



Rekursioni – Vazhdim



Rekursioni – Vazhdim



Rekursioni – Vazhdim


```
factorial
{ int n == 3   long answer == ?
  ...
  1> answer = 3 * PRIT factorial(2); }
  return answer;
}
```

```
factorial
{ int n == 2   long answer == ?
  ...
  2> answer = 2 * 1; }
  return answer;
}
```

Vizatimi i figurave rekursive

Algoritmi për vizatimin e një figure rekurzive vezësh:

1. Vizato figurën në prapavijë, të skaluar me madhësi paksa më të vogël sesa figura në paravijë. (Në qoftë se figura në prapavijë ka madhësinë zero, atëherë mos u merr me të.)
2. Vizato vezën në paravijë.
3. Vizato një kornizë përreth tërë figurës.

Vizatimi i figurave rekursive – Vazhdim

```
import javax.swing.*;
import java.awt.*;
/** Afishon figurën rekursive të një veje. */
public class RecursivePictureWriter extends JPanel
{ private double scale = 0.8;
  private int width = 400;
  private int height = 200;
  private int firstEgg = 150;
  /** Krijon dritaren. */
  public RecursivePictureWriter()
  { JFrame frame = new JFrame();
    frame.getContentPane().add(this);
    frame.setTitle("Vizatuesi rekursiv i vezëve");
    frame.setSize(width, height);
    frame.setBackground(Color.white);
    frame.setVisible(true);
  }
```

Vizatimi i figurave rekursive – Vazhdim

```
/** Vizaton figurën rekursive.
 * @param g - penda grafike */
public void paintComponent(Graphics g)
{ paintPicture(width, height, firstEgg, g); }
```

Vizatimi i figurave rekursive – Vazhdim

```
/**
 * @param right - skaji i djathtë i figurës
 * @param bottom - skaji i poshtëm i figurës
 * @param eggSize - madhësia e vezës që vizatohet
 * @param g - penda grafike */
private void paintPicture(int right, int bottom,
                          int eggSize, Graphics g)
{ int backgroundEggSize = (int)(eggSize * scale);
  if ( backgroundEggSize > 0 )
  { paintPicture((int)(right * scale), (int)(bottom * scale),
                 backgroundEggSize, g);
  }
  paintEgg(right - eggSize, bottom, eggSize, g);
  g.setColor(Color.black);
  g.drawRect(0, 0, right, bottom);
}
```

Vizatimi i figurave rekursive – Vazhdim

```
  /**
   * @param left - skaji i majtë i vezës
   * @param bottom - skaji i poshtëm i vezës
   * @param width - gjerësia e vezës
   * @param g - penda grafike */
  private void paintEgg(int left, int bottom,
                        int width, Graphics g)
  { int height = 2 * width / 3;
    int top = bottom - height;
    g.setColor(Color.pink);
    g.fillOval(left, top, width, height);
    g.setColor(Color.black);
    g.drawOval(left, top, width, height);
  }

  public static void main(String[] args)
  { new RecursivePictureWriter(); }
}
```

Vizatimi i figurave rekursive – Vazhdim

```
import javax.swing.*;
import java.awt.*;
/** Afishon një dru binar në formë ‘‘fushe vezësh.’’ */
public class EggFieldWriter extends JPanel
{ private int size = 20;
  private int totalLayers = 7;
  private int width = 600;
  private int height = 200;
  /** Krijon dritaren. */
  public EggFieldWriter()
  { JFrame frame = new JFrame();
    frame.getContentPane().add(this);
    frame.setTitle("Vizatuesi i një fushe vezësh");
    frame.setSize(width, height);
    frame.setBackground(Color.white);
    frame.setVisible(true);
  }
```

Vizatimi i figurave rekursive – Vazhdim

```
/** Vizaton figurën rekursive.
 * @param g - penda grafike */
public void paintComponent(Graphics g)
{ paintEggField(220, 200, totalLayers, g); }
```

Vizatimi i figurave rekursive – Vazhdim

```
/**
 * @param left - skaji i majtë i vezës së parë
 * @param baseline - skaji i poshtëm i vezës së parë
 * @param layer - numri i shtresave të vezëve për t'u vizatuar
 * @param g - penda grafike */
private void paintEggField(int left, int baseline,
                           int layer, Graphics g)
{ if ( layer > 0 )
  { int eggSize = size * layer;
    paintEggField(left + 2 * eggSize / 3,
                  baseline - eggSize / 3, layer - 1, g);
    paintEggField(left - eggSize / 2, baseline - eggSize / 3,
        layer - 1, g);
    paintEgg(left, baseline, eggSize, g);
  }
}
```

Vizatimi i figurave rekursive – Vazhdim

```
  private void paintEgg(int left, int bottom,
                        int width, Graphics g)
  { // ... sikur më parë
  }

  public static void main(String[] args)
  { new EggFieldWriter(); }
}
```