Preddiplomski studij:

3. godina

Diplomski studij

Razvoj usluga i aplikacija za operacijski sustav Android

1. Uvod u Javu 1

Objektno orijentirano programiranje u programskom jeziku Java

Ak.g. 2014./2015.

Zaštićeno licencom http://creativecommons.org/licenses/by-nc-sa/2.5/hr/

17.10.2014.

Creative Commons

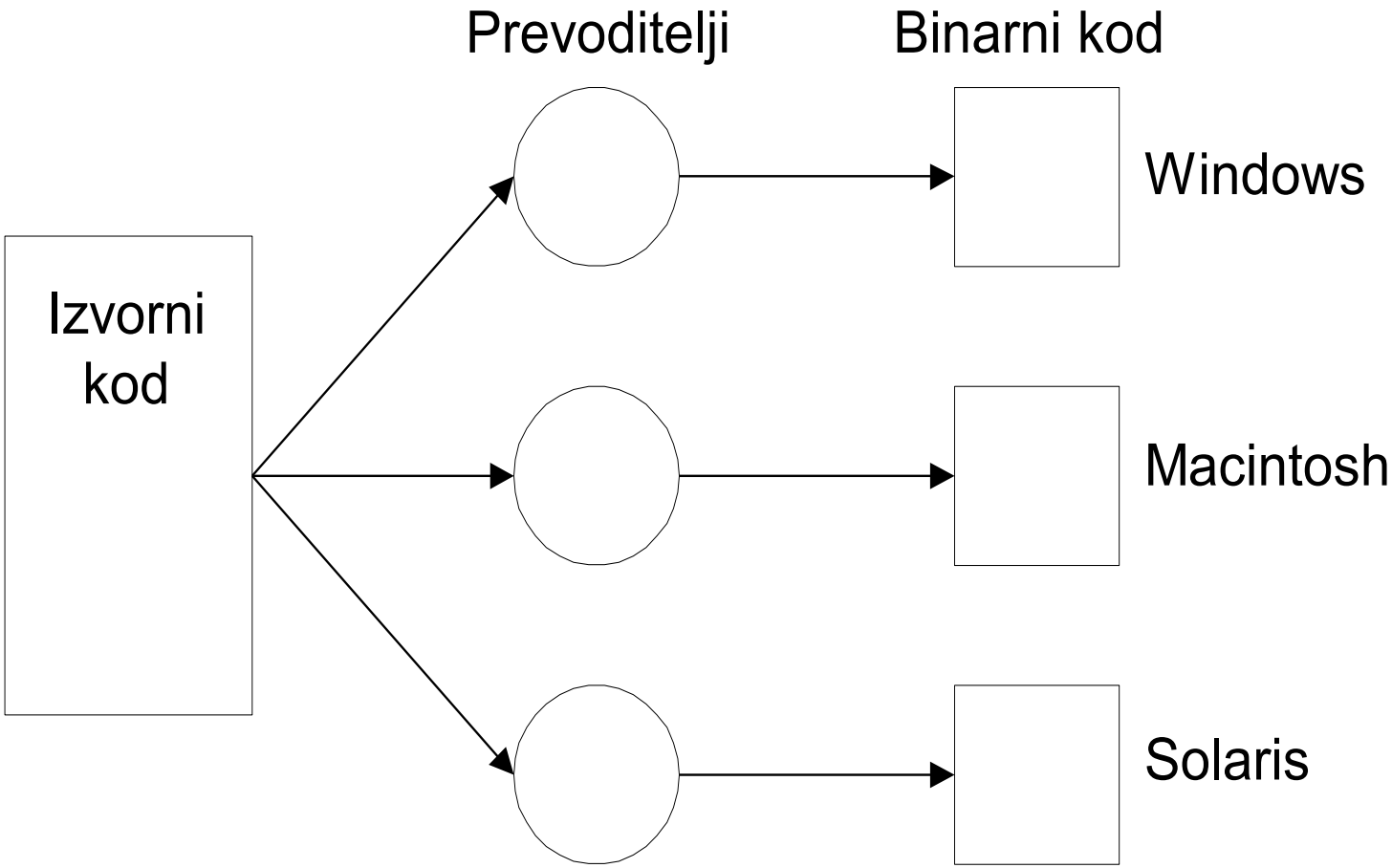
- **slobodno smijete:**
  - **dijeliti** — umnožavati, distribuirati i javnosti priopćavati djelo
  - **remiksirati** — prerađivati djelo
- **pod sljedećim uvjetima:**
  - **imenovanje**. Morate priznati i označiti autorstvo djela na način kako je specificirao autor ili davatelj licence (ali ne način koji bi sugerirao da Vi ili Vaše korištenje njegova djela imate njegovu izravnu podršku).
  - **nekomercijalno**. Ovo djelo ne smijete koristiti u komercijalne svrhe.
  - **dijeli pod istim uvjetima**. Ako ovo djelo izmijenite, preoblikujete ili stvarate koristeći ga, preradu možete distribuirati samo pod licencom koja je ista ili slična ovoj.

U slučaju daljnjeg korištenja ili distribuiranja morate drugima jasno dati do znanja licencne uvjete ovog djela. Najbolji način da to učinite je linkom na ovu internetsku stranicu.
Od svakog od gornjih uvjeta moguće je odstupiti, ako dobijete dopuštenje nositelja autorskog prava.
Ništa u ovoj licenci ne narušava ili ograničava autorova moralna prava.

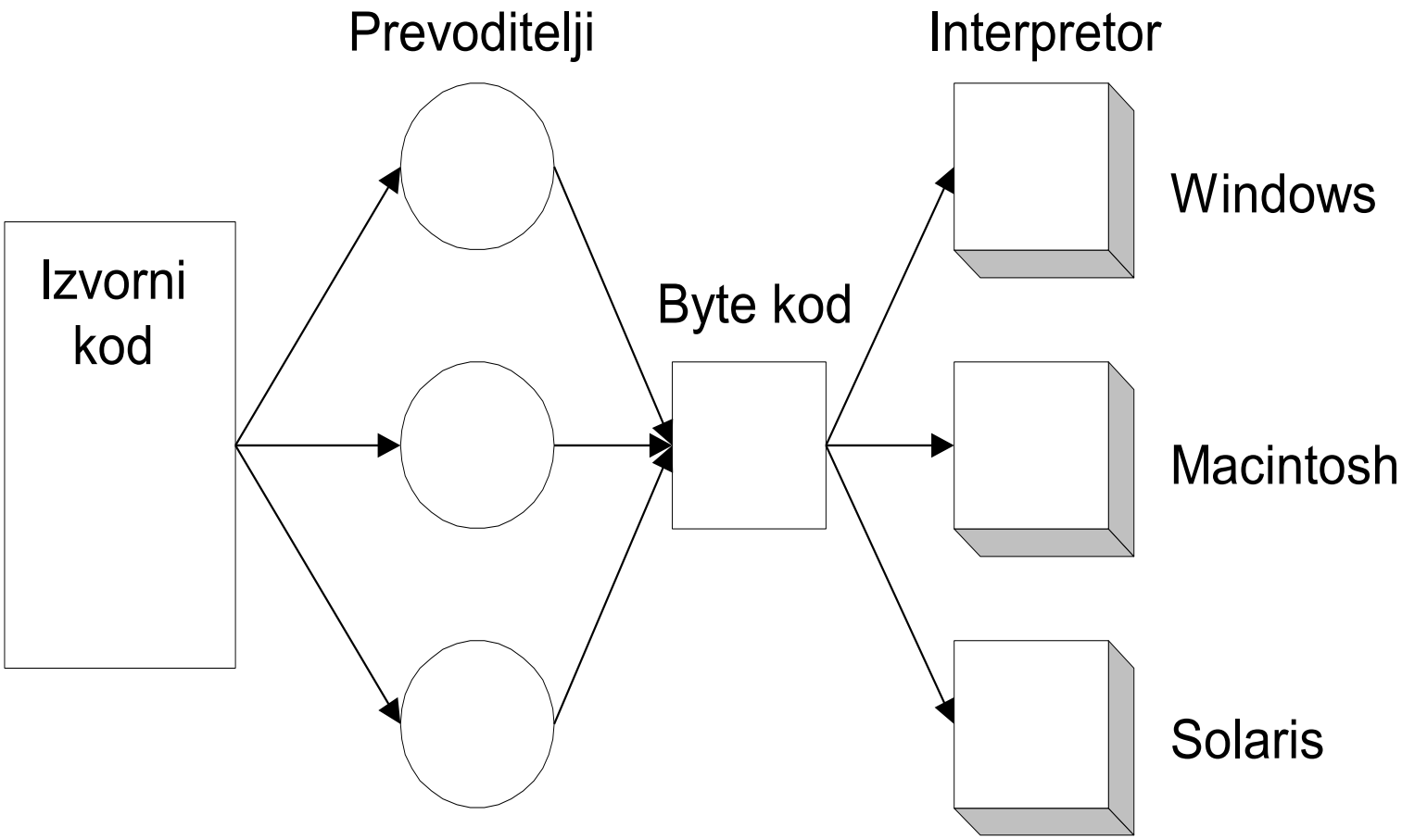
Tekst licencije preuzet je s http://creativecommons.org/.

Uvod u objektno orijentirano programiranje u programskom jeziku Java

Klasičan pristup izvođenja programa



Izvođenje programa pomoću virtualnog stroja



Prvi primjer

FirstExample.java:

```
public class FirstExample {
  public static void main(String[] args) {
        System.out.println("Ovo radi!!");
  }
}
```

prevođenje:

```
javac FirstExample.java
```

pokretanje:

```
java FirstExample
```

ispis:

```
Ovo radi!!
```

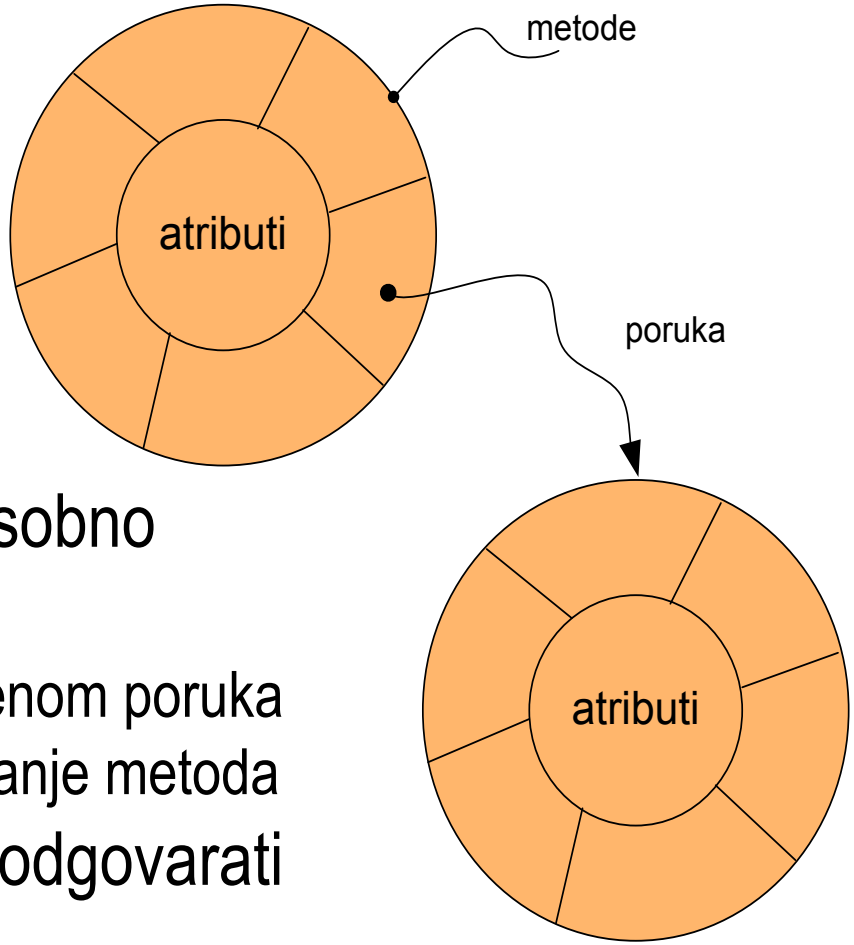
UML

- UML – *Unified Modeling Language*
  - učite ga na OPP-u
  - koristiti ćemo ga na vještini

- grafički jezik za:
  - vizualizaciju
  - specificiranje
  - konstrukciju i
  - dokumentiranje

objektno orijentiranih programskih rješenja

Objektni sustav



- objektni sustav = sustav međusobno povezanih objekata
  - međusobno komuniciraju razmjenom poruka
  - u Javi: razmjena poruka = pozivanje metoda
- programski objekt često može odgovarati "stvarnom" objektu

Ponašanje objekta, enkapsulacija

- osnovno načelo u OO-jezicima:
  - identifikator
  - stanje objekta skriveno
  - ponašanje objekta dostupno
    - ponašanje ovisi o trenutnom stanju
- skrivanje stanja = enkapsulacija
- Java:
  - ponašanje = koje vrijednosti vraćaju metode
    - metoda = objektova funkcija (javna)
  - stanje = trenutne vrijednosti atributa objekta
    - atribut = objektova privatna varijabla

Analiza zahtjeva

Brainstorming

- Za koga je sustav?
- Što će sustav raditi?
- Koji poslovne potrebe treba sustav podržavati?
- Koje informacije korisnik sustava treba imati?
- Podržava li sustav timski rad?
- Kako je najjeftinije to napraviti?
- Kako možemo to najbolje napraviti?

Aplikacija Quiz – zadatak

- Ideja:
  - Nastavnik slaže skup pitanja za studente
  - Student rješava ispit
  - Student može odgovarati na pitanja ispita
  - Postoje različite vrste pitanja:
    - Pitanja s kratkim odgovorima – npr. jedna riječ
    - Pitanja s više točnih odgovora – npr. a, c
    - Povezivanje pojmova – npr. a1,b2,c3

Primjer uporabe objekta

kreiranje objekta, zapis *reference* u varijablu

```
public class Main {

  public static void main(String[] args) {
    ShortAnswerQuestion question = new ShortAnswerQuestion(
        "U kojem jeziku se programira za Android?", "Java");

    System.out.println(question.getText());

    System.out.println("C: " + question.isCorrectAnswer("C"));
    System.out.println("java: " + question.isCorrectAnswer("java"));
  }
}
```

pozivi metoda nad objektom

Ispis programa:

```
U kojem jeziku se programira za Android?
C: false
java: true
```

Klasa `ShortAnswerQuestion`

```
public class ShortAnswerQuestion {
  private String text;
  private String correctAnswer;

  public ShortAnswerQuestion(String text, String
   correctAnswer) {
    this.text = text;
    this.correctAnswer = correctAnswer;
  }

  public String getText() {
    return text;
  }

  public boolean isCorrectAnswer(String answer) {
    return correctAnswer.equalsIgnoreCase(answer);
  }
}
```

atributi

konstruktor objekta

metode koje ispoljavaju ponašanje (povratna vrijednost ovisi o trenutnom stanju)

UML-ov dijagram klase `ShortAnswerQuestion`

```
public class ShortAnswerQuestion {
  private String text;
  private String correctAnswer;

  public ShortAnswerQuestion(String text, String
   correctAnswer) {
    this.text = text;
    this.correctAnswer = correctAnswer;
  }

  public String getText() {
    return text;
  }

  public boolean isCorrectAnswer(String answer) {
    return correctAnswer.equalsIgnoreCase(answer);
  }
}
```

ShortAnswerQuestion
-text: String -correctAnswer: String
+ShortAnswerQuestion() +getText() : String +isCorrectAnswer() : boolean

Klasa

- ono što definiramo datotekom izvornog koda
- predložak za objekte
  - koje atribute i metode će imati objekt
- objekt = instanca svoje klase
- klasa je vrsta podatka, kao i `int` ili `float`

- atribut: privatna varijabla objekta
  - svi objekti iste klase imaju iste atribute
  - ali svaki ima svoje vrijednosti atributa

Druge namjene klase

- ne služe sve klase za instanciranje
- npr. klasa *Main*
  - ima samo posebnu metodu main
  - nema atribute
  - nema “normalne” metode
  - instanca te klase je neupotrebljiva
- služi za pokretanje programa

Čitljivost koda

- pretežno **mala** slova (*lowercase*)
- **praznine** se koriste zbog čitljivosti
  - **uvlačenje** ocrtava ustroj cijele datoteke koda
  - **razmaci** određuju ustroj izraza i izjava
  - **prazne linije** odvajaju blokove koda
- postoje standardi kodiranja

Identifikatori i “čarobni brojevi”

- na **engleskom** jeziku
- identifikatori
  - klase i sučelja počinju **velikim**, svi ostali **malim** slovom
  - **logičke** varijable i metode: `isValid, canOpen`
  - dohvaćanje vrijednosti objekta: **get** → `getArea, getRealPart`
  - postavljanje vrijednost objekta: **set** → `setRealPart`
- **neimenovane konstante u kodu**
  - primjer: duljina polja u zadatku je 10
  - **nejasan** kod (“Otkud tu 10?”)
  - teškoće u **promjeni** konstante
    - ponekad “10” znači nešto **drugo**
    - negdje piše “od 0 do 9”
  - **definirati** i **smisleno nazvati** konstantu:

```
private static final int ARRAY_SIZE = 10;
```

Klase i metode

- Klase
  - predstavlja **predmet**, imenicu
  - što je moguće jednostavnija
  - samo **jedna** namjena

```
public class ClassName
   extends SuperClass
   implements Interface1,
   Interface2
{
   // tijelo uvučeno za
   // jedan korak
}
```

- Metode
  - predstavlja djelovanje, glagol
  - poželjno jedan ekran, najdulje A4 **stranica**
  - pazite na dug niz argumenata
    - skupite **srodne** argumente u jednu klasu
    - suzite **funkcionalnost** metode

```
public void methodName(
      SomeClass argument
)
throws SomethingThrowable
{
   // tijelo uvučeno za
   // jedan korak
}
```

Softverska dokumentacija

- **dokumentacija** je informacija o proizvodu namijenjena korisniku
- **unutrašnja** i **vanjska**
- programer dokumentira (opisuje) svoj rad za:
  - samog **sebe (unutrašnja)**
  - (poznate) **članove tima (unutrašnja)**
  - (nepoznate) **buduće programere (unutrašnja)**
  - ljude koji će samo **koristiti (vanjska)**

Komentari (1)

- **pozicioniranje**: **prije** koda
- ponekad na kraju linije koda (*inline*) – samo ako je komentar **vrlo kratak** i takvim će ostati
- odaberite stil oblikovanja
  - minimalni trud pri pisanju
  - lako se može proširiti ili prepraviti
- jednolinijski komentar: `// komentar`
- vrste komentara:
  - izvedbeni komentari
    - **ponavljaju** kod, **objašnjavaju** kod, **sažetak** koda, opisuju **namjeru**
  - **javadoc** komentari
  - **iskomentirani** kod
  - **oznake** u kodu

Komentari (2)

- Komentari koji **ponavljaju** kod

```
// z je korijen iz zbroja kvadrata
z = Math.sqrt( x*x + y*y );
```

  - ne valja:
    - nema novih informacija
    - zakrčuje kod
- Komentari koji **objašnjavaju** kod
  - kad je kod **prekompliciran**
  - obično je bolje **popraviti** kod nego ga **objašnjavati**
  - često na osnovu pretpostavke da čitatelj ne zna programski jezik ili njegov dio
  - objašnjenja su nužna kod **optimiziranog** koda

Komentari (3)

- Komentari kao **sažetak** koda
  - **veći** dio koda se **sažeto opisuje**
  - na taj način se **brže** prolazi kroz kod
  - u jeziku domene **rješenja**
  - dobar postupak:
    - napisati sučelje metode
    - tijelo popuniti pseudokodom na prirodnom jeziku
    - zamijeniti svaku liniju pseudokoda sa stvarnim kodom
    - pseudokod pretvoriti u komentare
- Komentari koji opisuju **namjeru** autora
  - ako je iz koda jasno **što** se radi
  - treba odgovoriti **zašto** to autor radi
  - u jeziku domene **problema**

Komentari Javadoc

- mehanizam generiranja vanjske **API dokumentacije** na osnovu komentara u kodu
- počinju s `/**`, završavaju s `*/`
- pišu se prije deklaracije klase, varijable ili metode
- može se formatirati u HTML-u
- posebne oznake u komentaru počinju s "`@`"
  - za klase: `author`, `version`...
  - za metode: `param`, `return`, `throws`, `deprecated`...

Primjer Javadoc izvornog koda

```
/**
 * This is the Question which accepts only short answer. Short
 * answer is one word.
 *
 * @author Mario Kušek
 *
 */
public class ShortAnswerQuestion {
   /**
    * Text of question.
    */
   private String text;

   /**
    * Correct answer of the question.
    */
   private String correctAnswer;

...
```

Generirani HTML iz primjera (1)

ShortAnswerQuestion

file:///Users/mario/work/text/android_vjestina/workspace-predavanja/Predavanje1/doc/

Twitter Personal Posao Predmeti Rječnik ISVU FER – Imenik A-STORM News Šalji poruku

All Classes

AbstractQuestion
AMultipleChoiceQuestion
AShortAnswerQuestion
EMultipleChoiceQuestion
EShortAnswerQuestion
IMultipleChoiceQuestion
IShortAnswerQuestion
Main
MainInterface
MainQuiz
MainScanner
MainTwoInstances
MultipleChoiceQuestion
Question
Quiz
ShortAnswerQuestion

Package **Class** **Use** **Tree** **Deprecated** **Index** **Help**

PREV CLASS NEXT CLASS FRAMES NO FRAMES
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Class ShortAnswerQuestion

```
java.lang.Object
  └ ShortAnswerQuestion
```

```
public class ShortAnswerQuestion
extends java.lang.Object
```

This is the Question which accepts only short answer. Short answer is one word.

Author:
Mario Kušek

Field Summary

private java.lang.String	**correctAnswer** Correct answer of the question.
private java.lang.String	**text** Text of question.

Constructor Summary

ShortAnswerQuestion(java.lang.String text, java.lang.String correctAnswer) Constructs question with text and answer.

Method Summary

java.lang.String	**getText**()

Display a menu

Komentari (4)

- **iskomentirani** kod
  - privremeno izbačen dio napisanog koda
  - jednolinijski – bez razmaka
    - `//System.out.println( "i = " + i );`
  - višelinijski sa **`/*`** i **`*/`**
- komentari koji služe kao **oznake** u kodu
  - oznake samom sebi ili drugim članovima tima
    - dio koda nije dovršen
    - ne nalazi se u konačnoj verziji koda
    - alati podržavaju brzo pronalaženje takvog koda
  - primjer:

```
public static void main(String[] args) {
        // TODO Auto-generated constructor stub
}
```

Primjer s dvije instance (`MainTwoInstances`)

```
public static void main(String[] args) {
   ShortAnswerQuestion question1 = new ShortAnswerQuestion(
       "U kojem jeziku se programira za Android?", "Java");
   ShortAnswerQuestion question2 = new ShortAnswerQuestion(
       "U kojem jeziku se programira za iPhone?", "Objective-C");

   System.out.println(question1.getText());
   System.out.print("objective-c: " +
     question1.isCorrectAnswer("objective-c"));
   System.out.println(" java: " + question1.isCorrectAnswer("java"));

   System.out.println(question2.getText());
   System.out.print("objective-c: " +
     question2.isCorrectAnswer("objective-c"));
   System.out.println(" java: " + question2.isCorrectAnswer("java"));
 }
```

Ispis programa:

```
U kojem jeziku se programira za Android?
objective-c: false java: true
U kojem jeziku se programira za iPhone?
objective-c: true java: false
```

Primjer čitanja s tipkovnice (`MainScanner`)

```
public static void main(String[] args) {
 ShortAnswerQuestion question = new ShortAnswerQuestion(
     "U kojem jeziku se programira za Android?", "Java");

 System.out.println(question.getText());

 Scanner scanner = new Scanner(System.in);

 boolean isCorrectAnswer = false;

 while(!isCorrectAnswer) {
   String answer = scanner.nextLine();
   isCorrectAnswer = question.isCorrectAnswer(answer);
   if(isCorrectAnswer)
     System.out.println("Točno!!!!");
   else
     System.out.println("Netočno.");
 }
}
```

Izgradnja sustava objekata

kompozicija i agregacija

Povezivanje objekata u sustav

- objekt A sadrži referencu na objekt B
  - time je A povezan s B
  - može pozivati njegove metode

Klasa `Quiz` (1)

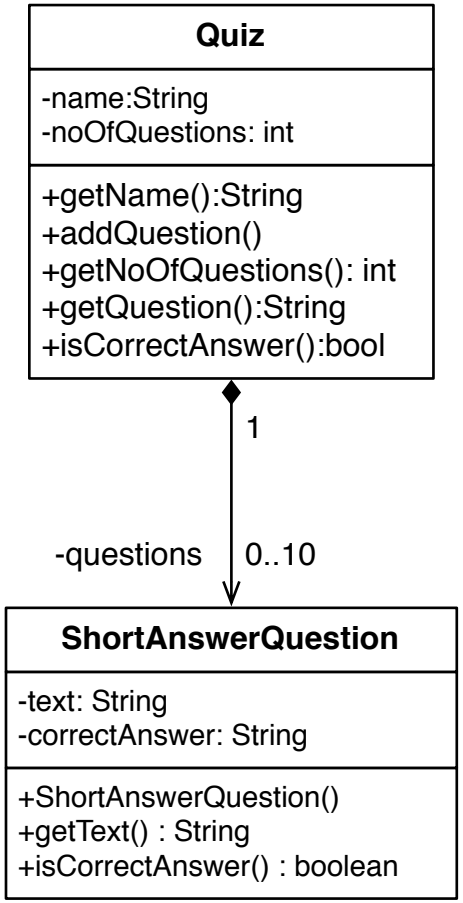
```
public class Quiz {
  private String name;
  private ShortAnswerQuestion[] questions = new ShortAnswerQuestion[10];
  private int noOfQuestions = 0;

  public Quiz() {
    name = "Initial Quiz";
  }

  public String getName() {
    return name;
  }

  public void addQuestion(String text, String answer) {
    questions[noOfQuestions] = new ShortAnswerQuestion(text, answer);
    noOfQuestions++;

    //TODO maksimalan broj pitanja
  }

  public int getNoOfQuestions() {
    return noOfQuestions;
  }
```

Klasa `Quiz` (2)

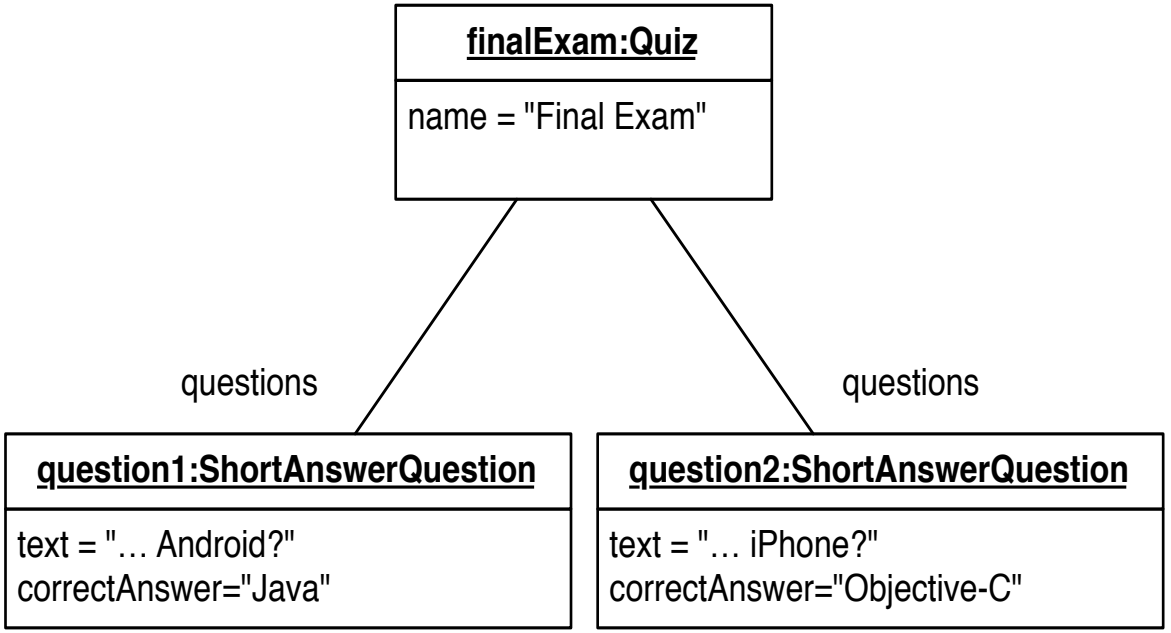
```
  public String getQuestion(int index) {
    return questions[index].getText();
  }

  public boolean isCorrectAnswer(int questionIndex, String answer) {
    return questions[questionIndex].isCorrectAnswer(answer);
  }
}
```

Kopozicija objekata

dijagram klasa



dijagram objekata



Primjer korištenja klase `Quiz`

```
public static void main(String[] args) {
  Quiz quiz = new Quiz();
  quiz.addQuestion("U kojem jeziku se programira za Android?", "Java");
  quiz.addQuestion("U kojem jeziku se programira za iPhone?",
     "Objective-C");

  Scanner scanner = new Scanner(System.in);
  int noOfCorrectAnswers = 0;
  for (int i = 0; i < quiz.getNoOfQuestions(); i++) {
    System.out.println(quiz.getQuestion(i));
    String answer = scanner.nextLine();
    if(quiz.isCorrectAnswer(i, answer)) {
      noOfCorrectAnswers++;
      System.out.println("Točno!");
    } else {
      System.out.println("Netočno.");
    }
  }

  System.out.println("Imali ste " + noOfCorrectAnswers + " točnih
   odgovora od ukupno " + quiz.getNoOfQuestions() + ".");
}
```

Konstruktor

- kreiranje objekta: potrebna je inicijalizacija
- dovođenje objekta u početno stanje
- to se obavlja konstruktorom
  - procedura koja se izvršava prilikom stvaranja objekta

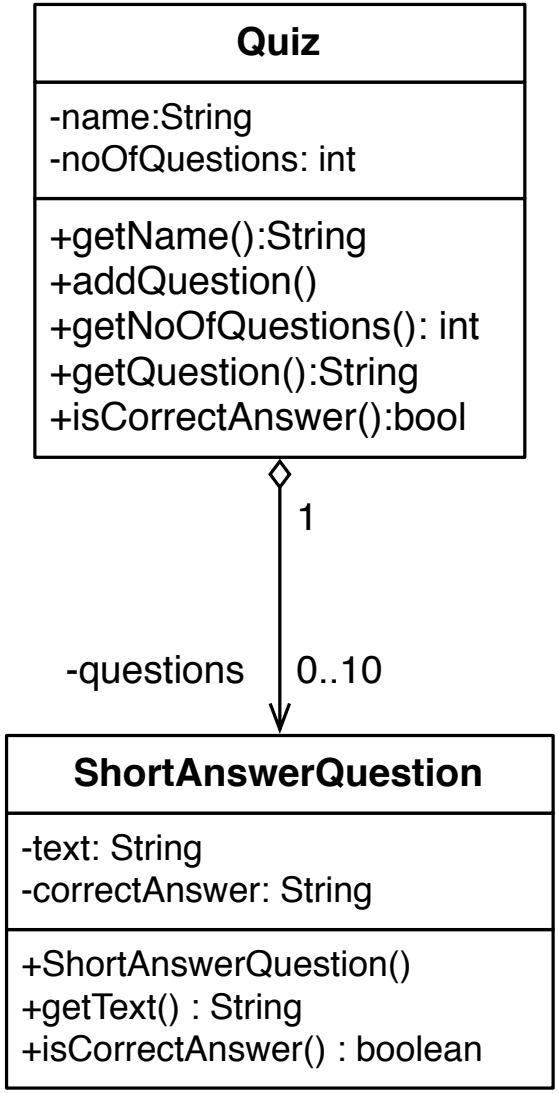
Repriza: klasa `ShortAnswerQuestion`

```
public class ShortAnswerQuestion {
  private String text;
  private String correctAnswer;

  public ShortAnswerQuestion(String text, String
   correctAnswer) {
    this.text = text;
    this.correctAnswer = correctAnswer;
  }

  public String getText() {
    return text;
  }

  public boolean isCorrectAnswer(String answer) {
    return correctAnswer.equalsIgnoreCase(answer);
  }
}
```

Konstruktor – klasa `Quiz`

```
public class Quiz {
  private String name;
  private ShortAnswerQuestion[] questions = new
   ShortAnswerQuestion[10];
  private int noOfQuestions = 0;

  public Quiz() {
    name = "Initial Quiz";
  }

  public Quiz(String name) {
    this.name = name;
  }

  public String getName() {
    return name;
  }
…
```

Agregacija - klasa `Quiz`

```
public class Quiz {
  private String name;
  private ShortAnswerQuestion[] questions = new ShortAnswerQuestion[10];
  private int noOfQuestions = 0;

  …

  public void addQuestion(String text, String answer) {
    questions[noOfQuestions] = new ShortAnswerQuestion(text, answer);
    noOfQuestions++;

    //TODO maksimalan broj pitanja
  }

  public void addQuestion(ShortAnswerQuestion question) {
    questions[noOfQuestions] = question;
     noOfQuestions++;

    //TODO maksimalan broj pitanja
  }
```

UML-ov dijagram – *agregacija*