TDP007 - Tenta

2024-06-07

Regler

- All kod efterrättas på denna tenta
- Vid toalettbesök ska vakt informeras.
- All form av kontakt mellan studenter under tentamens gång är strängt förbjuden.
- Böcker och anteckningssidor kan komma att granskas av assistent, vakt eller examinator under tentamens gång.
- Frågor om specifika uppgifter eller om tentamen i stort ska ställas via tentasystemets kommunikationsklient.
- Systemfrågor kan ställas till jourhavande genom att räcka upp handen
- Endast uppgifter inskickade före tentamenstidens slut rättas.

Hjälpmedel	En Rubybok (exempelvis the pickaxe) Ett A4-ark med egna anteckningar Tillgång till webresurser: ruby-docs, rubular och tidig version av kursboken

Information

Betygssättning vid tentamen

Tentamen består av ett antal uppgifter på varierande nivå. Uppgifter som uppfyller specifikationen samt följer god sed och konventioner krävs för maxpoäng på en uppgift. Avvikelser från ovanstående ger avdrag.

Tentan består av uppgifter (några indelade i deluppgifter) som totalt kan ge 50 poäng.

Ruby-docs

På tentan har du tillgång till referenssidorna på `https://ruby-doc.org/` (både core och std-lib finns tillgänglig) via webbläsaren.

Rubular

På tentan har du tillgång till sidan `https://rubular.com/`.

Givna filer

Eventuella givna filer finns länkade från kurssidan.

Avslutning

När du skickat in alla uppgifter och känner dig färdig kan du stänga tetaklienten och logga ut.

Uppgift 1 - Teori och grundläggande ruby (10p)

Uppgift 1a (5p)

Föreställ dig att du är lärare i den här kursen och vill skapa en uppgift som kontrollerar studenters kunskap om trädparsning i xml. Specifikt vill läraren i det här fallet kontrollera att studenterna kan nyttja xpaths på ett bra sätt. Uppgiften studenterna får kommer gå ut på att läsa ut data ur en xml-fil, men hur ska den xml-filen se ut? Ge exempel på 4 viktiga aspekter av utformningen av denna fil och ge en kort beskrivning av hur varje aspekt är viktig för att testa någon del av studenternas kunskap om xpath.

Uppgift 1b (5p)

Den här uppgiften testar att du förstår ett antal grundläggande begrepp inom objektorientering i allmänhet och Ruby i synnerhet. Skriv fullständig och körbar kod enligt följande långrandiga specifikation.

1. Skapa en klass Person som har två instansvariabler name och year. Instansvariablerna ska vara läsbara, men inte skrivbara. Ge klassen en konstruktor (som alltså anropas när vi kör new) som tar två parametrar motsvarande dessa två instansvariabler. Konstruktorn ska spara parametrarnas värden i instansvariablerna, som är tänkta att innehålla en sträng respektive ett heltal.

2. Skapa en klass Student som utökar klassen Person. Den ska ha ytterligare en instansvariabel points som är läsbar men inte skrivbar. Konstruktorn ska ta tre parametrar, motsvarande samtliga tre instansvariabler. Initieringsfunktionen för superklassen (Person) ska anropas för att initiera två av instansvariablerna, och den tredje initieras direkt i aktuell klass. Klassen Student ska dessutom ha en metod add som tar ett heltal som parameter och lägger till detta till det aktuella poängvärdet.

Uppgift 2 - rdparse (10p)

I den givna filen `example.css` finns ett exempel på en css-fil. Ditt jobb är att utöka rdparse med en ny parser som kan parsa css-filer och returnera en `Hash` med datan som finns i filen, se körexemplet för hur denna behållare ska struktureras.

Grammatiken som krävs för att i helhet täcka in css-formatet är förvånansvärt omfattande. Vi kommer därför i denna uppgift arbeta utifrån en förenklad variant. I denna variant kan en css-fil innehålla 2 saker: Kommentarer och Regler. En kommentar är godtyckliga tecken som förekommer mellan snedsträck stjärna och stjärna snedsträck `/* KOMMENTAR */`. Dessa kan kastas bort vid parsning.

En regel börjar med ett namn på element följt av en egenskapslista. Egenskapslistan innesluts av måsvingar `{ EGENSKAPSLISTA }`. Varje egenskap i egenskapslistan har ett namn och ett värde, dessa skiljs åt med ett kolon `NAMN: VÄRDE`. Varje egenskap avslutas med ett semikolon `NAMN: VÄRDE;`. Varken namn eller värden kan innehålla någonting annat än bokstäver eller siffror. Alla värden kan tolkas som strängar. Om ett namn förekommer i fler än en regel så ska alla egenskaperna som finns tillskrivas det namnet.

I exemplet nedan är alltså första raden en kommentar (som kan ignoreras i parsern). Sedan kommer en regel som gäller element av typen `nav`. Egenskapslistan för element av dessa typer innehåller två egenskaper, `list-style` och `padding`.

```
/* Navigation styling */
nav {
    list-style: none;
    padding: 0;
}
```

Körexempel (med hela filen example.css):

```
  {
    "header"=>{"align"=>"center", "padding"=>"20px", "color"=>"red"},
    "nav"=>{"padding"=>"0", "style"=>"none", "display"=>"inline"}
  }
```

Uppgift 3 - DOM-Parsning av XML(10p)

I den givna filen `books.xml` finns en samling av böcker. Det finns rollspelsböcker och vanliga böcker. I den givna filen uppgift3.rb finns ett påbörjat huvudprogram. Ditt jobb är att implementera funktionen som anropas i slutet av huvudprogrammet.

Funktionen ska använda en trädparsern för att hitta alla rollspelsböcker som uppfyller kriterierna och skriva ut dessa, se körexemplet för formattering. Kriterierna som finns i samband med det givna funktionsanropet är:

1. En lista av kategorier som boken kan ingå i, i det här fallet `Horror` och `Fantasy`
2. Ett årtal som boken måste vara släppt senare än, i det här fallet `2010`
3. Ett årtal som boken måste vara släppt innan, i det här fallet `2015`

Observera att alla böcker i kategorin `Horror` skrivs ut innan alla i kategorin `Fantasy`.

Körexempel:

```
Books in the Horror category:
Title: Call of Cthulhu Seventh Edition, Year: 2014, Publisher: Chaosium Inc.

Books in the Fantasy category:
Title: Dungeons and Dragons Players Handbook, Year: 2014, Publisher: Wi...
Title: The One Ring Roleplaying Game, Year: 2011, Publisher: Cubicle 7 E...
```

Utskriften trunkeras i körexemplet på grund av radlängden, du behöver inte göra det i din lösning.

Krav: Din lösning behöver vara robust nog för att hantera annorlunda ordning på elementen i xml-filen. Det är alltså inte säkert att rollspelsböckerna kommer innan eller efter de vanliga böckerna. Inte heller att varje genre kommer i samma ordning som i filen.

Uppgift 4 - SAX-Parsning av XML(10p)

I den givna filen `books.xml` finns en samling av böcker. Det finns rollspelsböcker och vanliga böcker. I den givna filen uppgift4.rb finns ett påbörjat huvudprogram. Ditt jobb är att implementera klassen som saknas (`BookFilter`) som ska vara en strömparser, som vanligt hittar du en mall för en strömparser `sax_example.rb`.

Din implementation av klassen ska skriva ut rollspelsböckerna som matchar kriterierna enligt körexemplet. Kriterierna som filtreras på är:

1. En lista av kategorier som boken kan ingå i, i det här fallet `Horror` och `Fantasy`
2. Ett årtal som boken måste vara släppt senare än, i det här fallet `2010`
3. Ett årtal som boken måste vara släppt innan, i det här fallet `2015`

Observera att alla böcker i kategorin `Horror` skrivs ut innan alla i kategorin `Post-Apocalyptic`.

```
Books in the Horror category:
Title: Call of Cthulhu Seventh Edition

Books in the Post-Apocalyptic category:
Title: Mutant: Year Zero
Title: The End of the World: Zombie Apocalypse
```

Krav: Din lösning behöver vara robust nog för att hantera annorlunda ordning på elementen i xml-filen. Det är alltså inte säkert att rollspelsböckerna kommer innan eller efter de vanliga böckerna. Inte heller att varje genre kommer i samma ordning som i filen.

Uppgift 5 - dsl (10p)

I den givna filen `dsl.rb` finns ett givet program skrivet i ett domänspecifikt språk, programmet är en konfigurationsfil. Justera den givna koden i `dsl_reader.rb` så att innehållet i filen läses in och sparas i en hash, se körexemplet.

Det givna programmet innehåller i 2 typer av rader. En typ av rad associerar data med en nyckel, detta är de första 4 raderna i programmet i `dsl.rb`. Först skrivs nyckeln följt av en lista av data som ska associeras med det namnet. Datan seppareras med kommatecken.

Den andra typen av rad är sådana som inleds med `delete` följt av en nyckel. Resultatet av att tolka denna rad ska vara att motsvarande nyckel tas bort från behållaren, observera att nyckeln `key` inte finns med i körexemplet.

Lägg till en lämplig print-funktion i din dsl-läsare och tillhörande anrop i huvudprogrammet så programmet skriver ut resultatet med lämpligt format (formateringen behöver inte följa körexemplet).

Körexempel:

```
{"dimensions"=>[1920, 1080],
 "acceleration"=>["software", "intel"],
 "path"=>["/home/ponha45/games/my_game"]}
```