Ordinarie tentamen för ID1004 Objektorienterad programmering, 12 januari 2012, 9-13

Denna tentamen examinerar 3.5 högskolepoäng av kursen.

Inga hjälpmedel är tillåtna.

Tentamen består av 12 frågor. Varje fråga kan ge mellan 1 och 4 poäng. Tentamensbetyget A-F bestäms i huvudsak av den ackumulerade poängsumman (korrekt besvarade frågor) enligt tabellen nedan. Examinator förbehåller sig möjligheten att vid gränsfall justera tentamensbetyget med hänsyn till en bedömning av helhetsintrycket.

Poäng	0-9	10-11	12-15	16-19	20-23	24-26
Tentamens-betyg	F	E	D	C	B	A

Läs noga igenom alla frågorna först och planera tiden. Lycka till!

Grundläggande syntaktiska konstruktioner och begrepp

1. (1p) Vad är det för problem (endast ett) med följande kommentar?

```
double tankVolym = 0.654; /* Storlek medium //
```

Svar: Kommentaren öppnas med /* och måste därför stängas med */. Alternativt kan kommentaren istället inledas med //

2. (1p) Vad är det för problem (endast ett) med dessa programsatser?

```
int summa = 1;
summa++;
int summa = 2 * summa;
```

Svar: Variabeln int summa deklareras två gånger.

3. (3p) Vilken text kommer att skrivas ut av denna kod?

```
String [] s = {"", "kah", "per", "na", "um", "go", " ", null};
System.out.println("Kejsaren heter " + s[3]+s[2]+s[1]+s[6]+s[3]+s[3]+s[5]);
```

Svar: Kejsaren heter naperkah nanago

Kommentar: Här gällde det att minnas att arrayindex börjar på noll och att texten ”Kejsaren heter ” också skrivs ut.

4. (4p) Vilken av de två alternativa villkorssatserna nedan är att föredra, och varför?

```
public int countNo(String [] answers) {
    int n = 0;
    for(String s : answers) {
        if ("no".equals(s)) { n++;} // Alt. 1
        if (s.equals("no")) { n++;} // Alt. 2
    }
    return n;
}
```

Svar: Om arrayen answers innehåller ett element som är null så kommer alternativ 2 att kasta en NullPointerException när man försöker referera s.equals(). Detta händer inte med alternativ 1 som därför är att föredra som mer robust.

Kommentar: Detta visade sig vara en svår fråga. Delpoäng har givits till de som förordat endera alternativet med en annan rimlig motivation. Trots flera förslag om motsatsen, så är även en strängkonstant i Java en instans av klassen String. Många har förordat att alt 2 är mer naturligt, men även alt 1 hade sina förespråkare med precis samma argument, så det är nog en vanesak.

Grundläggande datalogiska begrepp och relationer

5. (2p) Vad är det för problem med if-satsens villkor?

```
int age;
    // age tilldelas ett värde
if ((50 >= age) && (age >= 65)) {
    // övre medelålder
}
```

Svar: uttrycket blir aldrig sant pga &&-operatorn (båda sidor skall vara sanna); age kan inte samtidigt vara mindre än eller lika med 50 och större än eller lika med 65.

Kommentar: för denna typ av intervallvillkor kan man skriva termerna i stigande ordning från vänster till höger och alltid använda < eller <=. För åldrarna 50-65 blir det då:

```
if ((50 <= age) && (age <= 65)) {
```

6. (1p) Hur många tal skriver denna loop ut?

```
for (int n = 10; n >= 0; n--) {
  System.out.println(n);
}
```

Svar: 11 (10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0)

7. (1p) Vad finns det för alternativ kontrollstruktur till följande kod?

```
if (c == 'A') { … }
else if (c == 'B') { … }
else if (c == 'C') { … }
else if (c == 'D') { … }
else { … }
```

Svar: en switch/case –sats:

```
switch (c) {
case 'A':
  …
  break;
case 'B':
  …
  break;
case 'C':
  …
  break;
case 'D':
  …
  break;
default:
  …
  break;
}
```

8. (3p) I metoden nedan finns ett problem som Java-kompilatorn kommer att uppmärksamma. Granska koden och ange vilket problem du tror det är. Svaret ska innehålla både en beskrivning av problemet och var i koden det finns.

```
boolean contains (int wanted, int [] numbers) {
  boolean rtn = false;
  for (int n : numbers) {
    if (wanted == n) {
      return true;
      rtn = true;
    }
  }
  return rtn;
}
```

Svar: Java-kompilatorn kommer att klaga på unreachable code i raden ”rtn = true;” pga return-satsen på raden innan. Return-satsen gör att metoden aldrig kommer att exekvera nästa rad.

Kommentar: Många svar hävdade att det inte var tillåtet med mer än en return-sats i en metod. Det är dock inget som kompilatorn förhindrar, man kan ha flera. Det rekommenderas dock att man skriver sin kod med så få return-satser som möjligt. Om man tar bort return-satsen från metoden ovan så kommer metoden att fungera korrekt, men också att vara ineffektiv eftersom den alltid kommer att inspektera hela arrayen numbers, även om det sökta värdet hittas i arrayens första element. En mer effektiv implementation som inte gör mer arbete än nödvändigt är därför denna:

```
boolean contains (int wanted, int [] numbers) {
  boolean rtn = false;
  for (int n : numbers) {
    if (wanted == n) {
      rtn = true;
      break; // out of for
    }
  }
  return rtn;
}
```

Grundläggande objektorienteringsbegrepp

9. (1p) Vilka klasser måste innehålla metoden main nedan?

```
public static void main(String [] args) { … }
```

Svar: De klasser som innehåller ett huvudprogram, dvs den klass som programmet startar i.

10. (3p) Föreslå som mest tre mekanismer som Java-programmeraren har till sitt förfogande för att åstadkomma så kallad *inkapsling* (encapsulation). Motivera varje förslag.

Svar:

Klass-begreppet som medger uppdelning av koden i avgränsade delar

Åtkomstskydd av variabler (private, protected, default) som gör det möjligt att förhindra att variabler modifieras av kod utanför klassen eller dess package.

Åtkomstskydd av metoder (private, protected, default) som gör det möjligt att förhindra att metoder anropas av kod utanför klassen eller dess package.

Publika metoder (i kombination med private och protected) som gör det möjligt att definiera vilka metoder som extern kod kan anropa.

Deklarationen final som skyddar variabler från tilldelning eller förhindrar skuggning (åsidosättning, override) av metoder.

Interface-klasser som ger klassen möjlighet att implementera ett visst beteende utan exponera hur det är implementerat.

Undantagshantering (try-catch) som ger klassens metoder möjlighet att hantera fel utan att det syns uppåt.

11. (3p) Det går inte att skapa instanser av en abstrakt klass. Vad för nytta har man då av abstrakta klasser?

Svar: Nyttan är att den abstrakta klassen kan innehålla konkret kod som genom arv blir gemensam för alla underklasser, utan att samtidigt behöva specificera kod för metoder som saknar en generell implementation. Den abstrakta klassen fungerar därför som en mall för underklasserna. Underklasserna kan implementera egna versioner av de abstrakta klasserna och därigenom bli mindre abstrakta eller helt konkreta och instansierbara.

12. (3p) Förklara skillnaden mellan en *överlagrad* (overloaded) metod och en *skuggad* (overridden) metod.

Svar: en överlagrad metod finns i flera versioner som alla har samma namn men olika parametertyper. En skuggad (även kallad *åsidosatt*) metod ersätter en ärvd metod i en superklass genom att ha en lokal metod med samma namn och samma parametertyper (signatur) som den ärvda metoden.