CHALMERS

Institutionen för data- och informationsteknik

TENTAMEN

KURSNAMN	**Objektorienterad programmering, 7.5p**
PROGRAM:	**D2 (TKDAT-2)** **2012/2013, lp 1**
KURSBETECKNING	**DAT042**
EXAMINATOR	**Uno Holmer**
TID FÖR TENTAMEN	**Torsdagen den 17/1 2012, 08.30-12.30**
HJÄLPMEDEL	**Java API** (utdelas av skrivningsvakten)
ANSV LÄRARE	**Erland Holmström** **tel. 070-871 06 00** **besöker tentamen**
DATUM FÖR ANSLAG	**Senast den 8/2 2013** **datum för visning meddelas på kurshemsidan**
ÖVRIG INFORM.	**Betygsgränser: 3 - 24p, 4 - 36p, 5 – 48p. (max 60p)**

TENTAMEN: Objektorienterad programmering

Läs detta!

- *Uppgifterna är inte ordnade efter svårighetsgrad.*
- Börja varje uppgift på ett nytt blad.
- Skriv ditt idnummer på varje blad (så att vi inte slarvar bort dem).
- **Skriv rent dina svar. Oläsliga svar r ä t t a s e j!**
- Programkod som finns i tentamenstesen behöver ej upprepas.
- Programkod skall skrivas i Java 5 (eller senare) och vara indenterad och renskriven.
- Onödigt komplicerade lösningar ger poängavdrag.
- Omotiverad användning av klassvariabler och klassmetoder ger poängavdrag.
- Givna deklarationer, parameterlistor etc. får ej ändras.
- Läs igenom tentamenstesen och förbered ev. frågor.

I en uppgift som består av flera delar får du använda dig av funktioner klasser etc. från tidigare deluppgifter, även om du inte löst dessa.

Lycka till!

Uppgift 1

Välj **ett** alternativ för varje fråga! Garderingar ger noll poäng. Inga motiveringar krävs. Varje korrekt svar ger två poäng. *Besvara direkt i tesen!*

1. Givet klassen

```
public class Counter {
    private int i = 0;
    public Counter(int initialValue) { i = initialValue; }
    public Counter next1() { ++i; return this; }
    public Counter next2() { return new Counter(i+1); }
}
```

och kodavsnitten 1-3

```
List<Counter> list = new ArrayList<Counter>();

// 1
for ( Counter x : list )
    x = x.next1();

// 2
for ( Counter x : list )
    x = x.next2();

// 3
for ( Counter x : list )
    x.next1();
```

Vilket påstående är sant om vi antar att `list` inte är tom?

a. kodavsnitt 1 ändrar inga element i `list`, men 2 och 3 gör det.
b. kodavsnitt 2 ändrar inga element i `list`, men 1 och 3 gör det.
c. kodavsnitt 3 ändrar inga element i `list`, men 1 och 2 gör det.
d. inget av kodavsnitten ändrar något element i `list`.

2. Vilket påstående är falskt?
   a. testning kan påvisa förekomsten av fel i ett program.
   b. avlusning kan avslöja orsaken till fel i ett program.
   c. testning och avlusning är samma sak.
   d. regresionstestning bör utföras efter refaktorering.

3. Antag att `Super` är basklass till `Sub` och att man i `Sub` vill definiera en överskuggning av standardmetoden `equals` i `Object`. Vilken deklaration av parametern är korrekt?

```
public boolean equals(Object x)         // 1
public boolean equals(Super x)          // 2
public boolean equals(Sub x)            // 3
```

   a. bara 1
   b. bara 2
   c. 2 och 3
   d. bara 3

4. Betrakta följande javapaket och klasser:

```
package module1;

public class C1 {
    private int x;
    protected int y;
    public int z;
}

public class C2 extends C1 { ... }

public class C3 { ... }
```

```
package module2;
import module1.*;

public class C4 extends C1 { ... }

public class C5 { ... }
```

Låt C_i (v_1, … ,v_n) beteckna att variablerna v_1, … ,v_n är synliga i klassen C_i. Vilket av alternativen beskriver synligheten hos variablerna x, y och z i klasserna ovan?

a. C1 (x,y,z), C2(y,z), C3(z), C4(y,z), C5(z)
b. C1 (x,y,z), C2(y,z), C3(z), C4(z), C5(z)
c. C1 (x,y,z), C2(y,z), C3(y,z), C4(z), C5(z)
d. C1 (x,y,z), C2(y,z), C3(y,z), C4(y,z), C5(z)

5. Följande kodavsnitt avser att beräkna totala antalet heltal i en mapp av listor.

```
Map<String,List<Integer>> m =
    new HashMap<String,List<Integer>>();

...
int count = 0;
for ( String key : m.keySet() ) {
    List<Integer> l = m.get(key);
    count += l.size();
}
System.out.println(count);
```

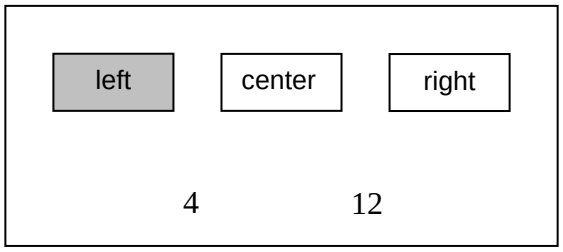
a. koden är inte säker, ett exekveringsfel kan uppkomma
b. deklarationen på första raden är inte typkorrekt
c. koden är felfri så långt man kan se av vad som visas ovan

(10 p)

Uppgift 2

Ett spel går till på följande sätt. Bakom en av tre skärmar döljer sig ett föremål. Spelaren skall gissa var föremålet befinner sig. Om den första gissningen är rätt får spelaren två poäng, annars ges en chans till. Är den andra gissningen rätt får spelaren en poäng, annars förlorar spelaren alla sina poäng. Spelaren får spela tio sådana omgångar. Maximalt antal poäng är alltså 20.

Uppgiften går ut på att konstruera ett program med grafiskt användargränssnitt för spelidén ovan. I fönstret skall finnas tre knappar som spelaren trycker på för att gissa var föremålet finns. En knapp som tryckts ner avaktiveras till nästa spelomgång. Endast tryck på aktiverade knappar har effekt. Under de tre knapparna i fönstret skall två uppgifter redovisas, antal omgångar kvar samt uppnådda poäng.



I exemplet ovan har spelaren tryckt ner den vänstra knappen, vilket var fel och får en chans till. Spelaren har totalt tolv poäng samt fyra omgångar kvar, inklusive den pågående.

Konstruera det grafiska gränssnittet och koppla i hop det med ett objekt av nedanstående klass. Till din hjälp finns klassen `GameState` som håller reda på spellogiken.

```
public class GameState {
    ...
    // score: register the player´s choice and update the score etc
    // parameter: guessedButton: a number in the range 1-3
    public void score(int guessedButton)

    // return the player´s current score
    public int getScore()

    // return the number of games left to play
    public int gamesLeft()

    // return true if ready for a new game, false ow
    public boolean isNewGame()
}
```

Textutmatning i fönstret sker enklast i JLabel-objekt. Likartad kod får anges schematiskt.

(12 p)

Uppgift 3

Klassen `ObjectCounter` håller reda på antalet objekt av subklasser till sig själv. Klassen skall ha metoden

```
public int getCount() {   // returnerar antalet objekt
```

Exempel:

```
public class Sub1 extends ObjectCounter { ... }
public class Sub2 extends ObjectCounter { ... }

ObjectCounter o = new ObjectCounter();
System.out.println(o.getCount());         // 1

Sub1 a = new Sub1();
System.out.println(a.getCount());         // 1

Sub1 b = new Sub1();
System.out.println(b.getCount());         // 2

Sub1 c = new Sub1();
System.out.println(c.getCount());         // 3

Sub2 d = new Sub2();
System.out.println(d.getCount());         // 1

Sub2 e = new Sub2();
System.out.println(e.getCount());         // 2
```

Skriv färdigt `ObjectCounter`.

Ledning: Metodanropet `obj.getClass().getName()` returnerar den dynamiska typen för `obj`. Använd en lämplig struktur för att lagra information om de olika subklasserna samt inför en lämplig konstruktor.

(8 p)

Uppgift 4

a) Vad skrivs ut av följande metodanrop? Motivera!

```
Base obj = new Sub();
obj.f1();                   // 1.
obj.f2();                   // 2.
obj.f3();                   // 3.

Sub obj2 = new Sub();
obj2.f1();                  // 4.
obj2.f2();                  // 5.
obj2.f3();                  // 6.
obj2.f4();                  // 7.
```

```
public class Base {
    public void f1() { System.out.print("Base.f1"); }
    public void f2() { System.out.print("Base.f2"); }
    public void f3() { System.out.print("Base.f3"); }
}

public class Sub extends Base {
    public void f1() { System.out.print("Sub.f1"); }
    public void f2() { super.f2();System.out.print("++Sub.f2"); }
    public void f4() { System.out.print("Sub.f4"); }
}
```

(7 p)

b) Vilka av raderna 1-4 är tillåtna och vilka ger kompileringsfel? En av klasserna går ej att kompilera, vilken? Motivera svaren!

```
Int  obj1 = new Int();          // 1
Base obj2 = new Base();         // 2
Sub1 obj3 = new Sub1();         // 3
Sub2 obj4 = new Sub2();         // 4

public interface Int {
    public void h();
}

public abstract class Base implements Int {
    public abstract void f();
    public void g() { ... }
}

public class Sub1 extends Base {
    public void f() { ... }
}

public class Sub2 implements Int {
    public void h() { ... }
}
```

(8 p)

Uppgift 5

En backup-demon är ett aktivt objekt som periodiskt sparar ett annat objekt till fil. Flera backup-demoner kan vara aktiva samtidigt och spara sina resp. objekt med olika inbördes tidsintervall. Uppgiften är att komplettera klassen `BackupDaemon`.

Inparametrar till konstruktorn är objektet som skall sparas, namnet på backup-filen, samt tidsintervallet i sekunder. Varje objekt av klassen skall exekveras i en egen tråd.
Exempel: Antag att klasserna `SomeClass` och `SomeOtherClass` är serialiserbara.

```
SomeClass o1 = new SomeOtherClass(), o2 = new SomeOtherClass();
new BackupDaemon( o1, "s1.back", 60 );
new BackupDaemon( o2, "s2.back", 300 );
```

`o1` sparas då en gång per minut, och `o2` var femte minut i respektive fil. Varje gång ett objekt sparats skall ett meddelande i stil med följande exempel skrivas ut.

```
Object written to s1.back
```

Implementera filhanteringen i en privat metod i klassen.

(8 p)

Uppgift 6

Antag att klassen B är intresserad av alla tillståndsförändringar som sker i A. När instans-variabeln i A ändras skall B skriva ut det nya värdet.

```
public class A {
    private int value = 0;

    public void compute(int x) {
        value += x;
    }
}

public class B {
    private A theAObject;

    public B(A anAObject) {
        theAObject = anAObject;
    }
}
```

Modifiera klasserna så att designmönstret Observer realiseras.

(7 p)