TENTAMEN: Objektorienterad programmering

Läs detta!

- *Uppgifterna är inte ordnade efter svårighetsgrad.*
- Börja varje uppgift på ett nytt blad (skriv inga lösningar i tesen!).
- Numrera och ordna bladen i uppgiftsordning.
- Skriv din tentamenskod 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.
- Du behöver ej skriva importdeklarationer.
- 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.

1. Givet klassen `Num`

```
public class Num {
    private int x;
    public Num() { x = 0; }
    public void set( int x ) { this.x = x; }
    public int get() { return x; }
}
```

vad skriver kodavsnittet nedan ut?

```
ArrayList<Num> list = new ArrayList<Num>();
Num n = new Num();

for ( int j = 0; j < 3; j++ )
    list.add( n );

int k = 1;
for ( Num x : list )
    x.set( k++ );

for ( Num x : list )
    System.out.println( x.get() );
```

a. `0 0 0`
b. `3 3 3`
c. `1 2 3`

2. Vilket påstående är korrekt om följande Javaklass?

```
public class C {
    private int x = 123;
    private static int y = 456;

    public void f() { ... }
    public static void g() { ... }
};
```

a. `f` får anropa `g` och ändra `x` och `y`
b. `g` får anropa `f` och ändra `x` och `y`
c. `g` får ändra `x`
d. `g` får läsa av men ej ändra `x`
e. `g` får anropa `f`

3. Om `A` är basklass till `B` och `B` basklass till `C` och variablerna `a`, `b` och `c` har typerna `A`, `B` resp. `C`, vilken tilldelning är då tillåten i Java?
   a. `b = a;`
   b. `c = b;`
   c. `b = c;`
   d. `c = a;`

4. En klass som präglas av hög kohesion

   a. har många abstrakta metoder
   b. har inga abstrakta metoder
   c. kan ha många olika ansvarsområden
   d. har ett väl avgränsat ansvarsområde

5. En av deklarationerna ger ett typfel, vilken?

   a. `ArrayList<LinkedList> l = new ArrayList<LinkedList>();`
   b. `List<LinkedList> l2 = new ArrayList<LinkedList>();`
   c. `ArrayList<List> l = new ArrayList<List>();`
   d. `List<LinkedList> l = new ArrayList<List>();`
   e. `List<List> l = new ArrayList<List>();`

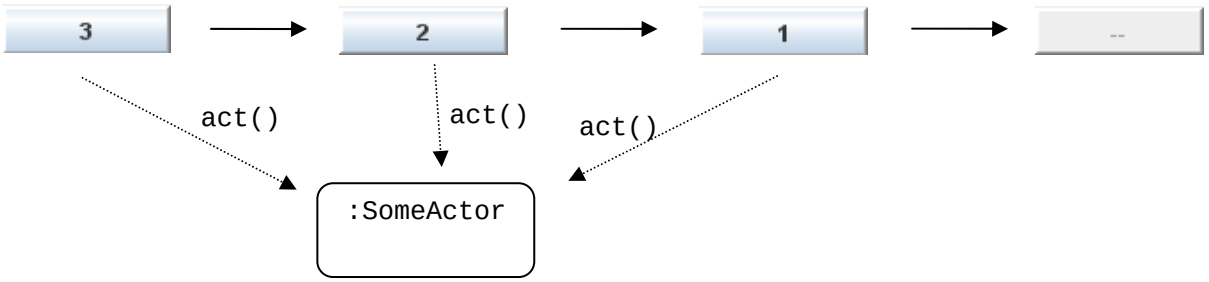
(10 p)

Uppgift 2

En N-knapp minns hur många gånger den tryckts ned. Vid varje knapptryckning anropas metoden `act()` i ett objekt av typen `Actor`:

```
public interface Actor {
    void act();
}
```

Efter N nedtryckningar inaktiverar knappen sig själv. I knappen visas antalet möjliga återstående tryckningar som leder till anrop av `act`. Efter sista tryckningen visas en ”--”-sträng i den inaktiverade knappen. Nedan visas hur en ”3-knapp” förändras vid tre nedtryckningar. (*OBS! Det är inte tre olika knappar*.)



Definiera klassen `CounterButton` med arv från lämplig Swing-klass. Låt klassen vara sin egen lyssnare. Om klassen `SomeActor` implementerar gränssnittet ovan skall man t.ex. kunna skapa 3-knappen ovan så här

```
new CounterButton(3,new SomeActor(...));
```

(10 p)

Uppgift 3

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"); }
}
```

(5 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() { ... }
}
```

(5 p)

Uppgift 4

a)
Antag att man vill definiera en klass `Point` för punkter i planet

```
public class Point
{
    public static final double tolerance = 0.001;
    private double x;
    private double y;
    ...
}
```

samt överskugga metoden `equals` så att två punkter betraktas som lika om avståndet mellan dem inte överstiger `tolerance`. Är det lämpligt att definiera likhet på detta sätt? Motivera i så fall varför, eller visa med ett motexempel en situation där `equals` skulle ge ett icke önskvärt resultat.

(2 p)

b)
En tidningsdistributör har ett prenumerantregister med person- och adressuppgifter enligt nedan.

```
public class Prenumerant
{
    private String name;
    private String phone;
    private String address;
    ...
}
```

Definiera en `equals`-metod i `Prenumerant` så att två prenumerantobjekt är lika om namnen är lika och personerna bor på samma adress. Metoden skall definieras så att den inte kan överskuggas vid eventuellt arv från `Prenumerant`. För full poäng krävs att metoden kan hantera parametrar av godtycklig typ samt `null`-värden.

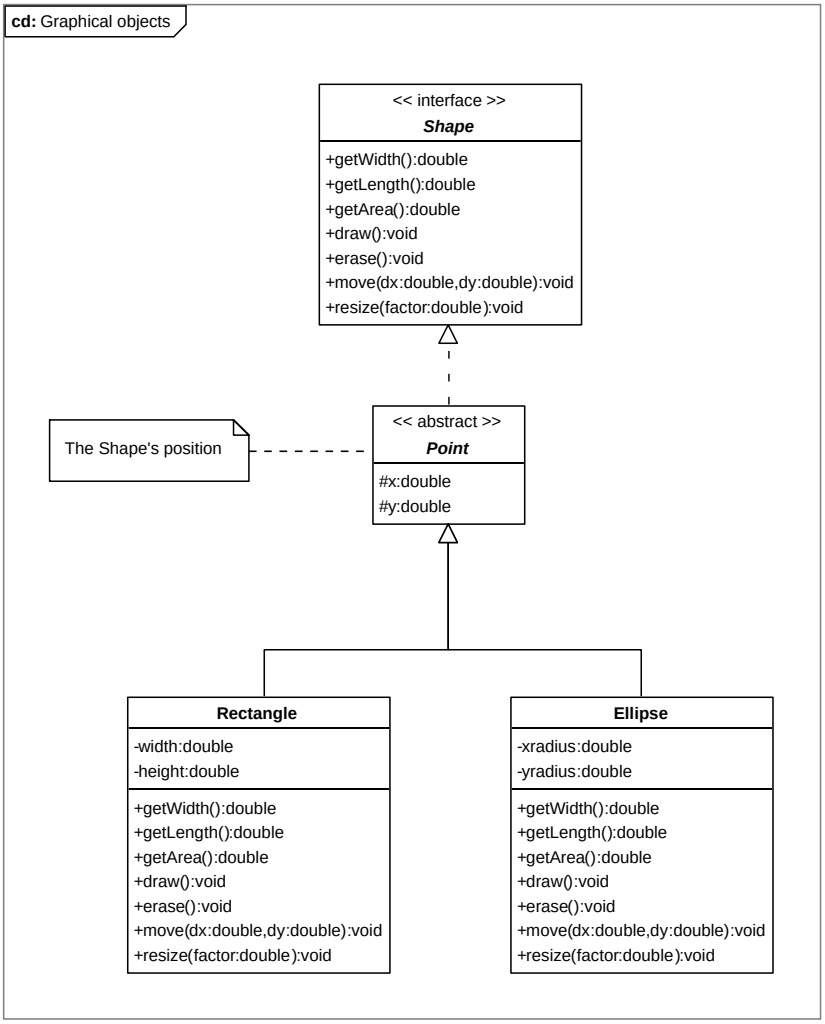
(4 p)

c)
Definiera metoden `hashCode` i `Prenumerant` på lämpligt sätt.

(2 p)

Uppgift 5

Antag att man har tillgång till följande klasser för att representera geometriska figurer



Klasserna är givna och får inte ändras. Antag att man vill kunna addera en tredje dimension till rektangel- och ellipsobjekt så att de får en volym. De skall alltså även ha en höjd. Inför klassen `Volume`. Klassen skall bl.a. ha metoden

```
public double getVolume()
```

Visa hur designmönstret Decorator kan användas för att lösa problemet, utan att ärva från rektangel- eller ellipsklassen.

a) Komplettera klassdiagrammet med klassen `Volume`, samt visa vilka instansvariabler och metoder klassen skall ha. Visa dessutom hur den är relaterad till andra klasser i diagrammet. Det räcker om de befintliga klasserna anges med namn i det nya diagrammet.
(3 p)

b) Implementera klassen `Volume` i Java.
(6 p)

c) Ge ett kodexempel som skapar en kub och skriver ut dess volym.
(1 p)

Uppgift 6

Vid en viss högskola används följande metod för att kommunicera tentamensresultat till studenterna. En databas innehåller information om studenternas personnummer (cid), namn, adressinformation, registrerade kurser, samt uppnådda poäng. Databasen läses sekventiellt via en objektström. Tentamensresultat rapporteras i textfiler, en per tentamen, av den rättande läraren. Varje rad i en sådan fil består av tre kommaseparerade fält med personnummer, skrivningspoäng, samt betyg. Till exempel `DAT041.txt` från augustitentan 2039

```
871293-3625,39,4
851923-6529,51,5
860342-8732,27,3
```

Uppgiften är att konstruera delar av ett program som från en sådan databas och en resultatfil producerar e-brev till de på kursen registrerade studenter som skrivit tentan. Om studenterna som gick upp på tentan ovan var

Namn	Personnummer	E-postadress	Registrerade kurser
Bosse Lindspjut	871293-3625	bosse@student.chapman.se	LET123, DAT041
Lena Ladok	851923-6529	ladokl@student.chapman.se	DAT050, DAT055
Katrin Borin	860342-8732	katrin@student.chapman.se	DAT041

så skall följande två e-brev skickas

```
To: bosse@student.chapman.se
Subject: Exam result DAT041
Hello Bosse Lindspjut!
Your marks: 39, grade: 4
```

```
To: katrin@student.chapman.se
Subject: Exam result DAT041
Hello Katrin Borin!
Your marks: 27, grade: 3
```

Lena Ladok var ju inte registrerad på kursen och får därför, i väntan på utredning, inget brev. Följande färdiga klasser finns:

```
public class ContactInformation implements Serializable {
    private String street;
    private String areaCode;
    private String city;
    private String email;

    public ContactInformation(String street,String aareaCode,
                              String city,String email)
    {
        this.street = street;
        this.areaCode = areaCode;
        this.city = city;
        this.email = email;
    }

    public String getStreet() { return street; }
    public void setStreet(String newStreet) { street = newStreet; }
    public String getCode() { return areaCode; }
    public void setCode(String newCode) { areaCode = newCode; }
    public String getCity() { return city; }
    public void setCity(String newCity) { city = newCity; }
    public String getEmail() { return email; }
    public void setEmail(String newEmail) { email = newEmail; }
}
```

```
public class StudentRecord implements Serializable {
    private String cid;
    private String name;
    private ContactInformation contact;
    private Set<String> courseRegistrations;
    private int credits;

    public StudentRecord(String cid,String name,ContactInformation contact) {
        this.cid = cid;
        this.name = name;
        this.contact = contact;
        courseRegistrations = new HashSet<String>();
        credits = 0;
    }

    public String getCid() {
        return cid;
    }
    public String getName() {
        return name;
    }
    public void addCourseRegistration(String code) {
        courseRegistrations.add(code);
    }
    public boolean isRegistred(String code) {
        return courseRegistrations.contains(code);
    }
    public void setContact(ContactInformation contact) {
        this.contact = contact;
    }
    public ContactInformation getContact() {
        return contact;
    }
    public void setCredits(int credits) {
        this.credits = credits;
    }
}

public class ExamGrade {
    private int marks;
    private String grade;

    public ExamGrade(int marks,String grade) {
        this.marks = marks; this.grade = grade;
    }
    int getMarks() { return marks; }
    String getGrade() { return grade; }
}

public class MailServer { // Singleton class
    ...
    public static MailServer getInstance()
    ...
    public void sendmail(String to,String subject,String message)
    ...
}
```

Klassen `ResultProcessor` skall vid instansieringen analysera en databasfil och en tentamensresultatfil och sända e-post enligt ovanstående beskrivning.

```
public class ResultProcessor {

    public ResultProcessor(String databaseFileName,String resultsFileName) {

        ...
    }
    private void processFiles() throws IOException {

        ...
    }


    // given
    String getNameBase(String fileName) // return the base part of fileName
}
```

Tips: Genom att först läsa in textfilen med tentamensresultaten till en lämplig datastruktur blir det senare enkelt att ta fram tentamensresultatet för ett visst personnummer. Använd metoden `split` i strängklassen (se bilaga) för att plocka ut de olika fälten i en kommaseparerad textrad, `regexp` kan t.ex. vara `",`". Detta görs lämpligen i en privat metod. Undantaget `IOException` bör kastas om något fält fattas. Du kan anta att det sista objektet i databasen är `null`, vilket kan användas för att hitta filslutet.

(12 p)