Tentamen Programmering fortsättningskurs DIT950

Joachim von Hacht

Datum: 2013-08-28

Tid: 08.00-12.00

Hjälpmedel: Engelskt-Valfritt språk lexikon

Betygsgränser:

- U: -23
- G: 24-43
- VG: 44-60 (max 60)

Lärare: Någon (besöker ca 09.30 och 11.00), 0707/311066

Granskning: Tentamen kan granskas på studieexpeditionen. Vi ev. åsikter om rättningen eposta mig och ange noggrant vad du anser är fel så återkommer jag.

Instruktioner:

- För full poäng krävs ett läsbart, begripligt och heltäckande svar. Generellt 1p för varje relevant aspekt av problemet, konkreta kodexempel kan ge mer.
- Det räcker med enbart relevanta kodavsnitt, övrig kod ersätts med "..." (aldrig import, hjälpklasser såsom Main, main-metod, etc....)
- Oprecisa eller alltför generella (vaga) svar ger inga poäng. Konkretisera och/eller ge exempel. Det är aldrig någon risk att vara övertydlig!

LYCKA TILL...

1. Vad avses med (ingen uttömmande förklaring krävs); 4p

   a) Statisk metod.

   b) Sidoeffekt. Ge ett kod exempel!

   c) Explicit typomvandling (casting).

   d) Failure atomicity.

2. Givet klasserna och gränssnittet nedan (ev. vänd sida). 8p

   a) Rita ett UML klassdiagram för dessa.

   b) Ange för varje kommentar a)-f) i koden nedan om kodstycket är korrekt eller ej. Om det ej är korrekt specificera om det är compile-time eller runtime-fel. Om det är korrekt, vad skrivs ut?

```
// a
A a = new B();
a.doIt();

// b
IX x = new D();
B b = (B) x;
b.doIt();
// c
A a = new C();
a.doIt();
// d
A a = new B();
C c = (C) a;
c.doIt();
// e
IX x = new D();
x.doOther();
// f
IX x1 = new IX(){
   @Override
   public void doOther() {System.out.println("IX doIt");}
};
C c1 = (C) x1;
c1.doIt();

// ------ Types -------------------
public interface IX { public void doOther(); }

public static class A {
```

```
    public void doIt() { System.out.println("A doIt");}
}

public static class B extends A {

    public void doIt() { System.out.println("B doIt";}
}

public static abstract class C extends A implements IX {

    public void doYetOther() { System.out.println("C doYetOther");}
}

public static class D extends C {

    public void doOther() { System.out.println("D doOther");}
}
```

3. Skapa en icke-muterbar klass för planeter. Klassen skall ha följande attribut; 4p

```
double mass;
String name;
Date dateOfDiscovery;
```

   Obs! Att klassen Date är muterbar (java.util.Date).

4. Antag att du har klassen; 6p

```
public class A {
    private int id = 99;
    private List<String> strs = new  ArrayList<String>();
  :
  :
}
```

   Implementera en equals-metod för klassen. Metoden skall överskugga (override) Objects equals-metod. Motivera din implementation och ange eventuella andra möjligheter.

5. Implementera att program som summerar värdena i ett träd med noder enligt nedan. För full poäng krävs en svansrekursiv variant. Någon eventuell hjälpklass är tillåten. 6p

```
public class Node {
```

```
    private final int value;
    private final Node left;
    private final Node right;
    public Node(int value, Node left, Node right) {
        super();
        this.value = value;
        this.left = left;
        this.right = right;
    }
    public int getValue() { return value; }
    public Node getLeft() { return left; }
    public Node getRight() { return right; }
}
```

6. Betrakta klasserna nedan. 8p

```
public class Main {
  public static void main(String[] args) {
    Sub s = new Sub();
    s.m();
    System.out.println(s.getX());
    s.n();
    System.out.println(s.getX());
  }
}

public class Base  {
  protected int x = 0;
  protected void m() { x++; }
  protected void n() {
    x += 2;
    //m();       // <--------
    }
  public int getX(){ return x; }
}

public class Sub extends Base {
  protected void m() { x += 5;}
  protected void n() {super.n();}
}
```

a) Vad kommer att skrivas ut i main;

b) Antag att utvecklaren av basklassen modifierar basklassen genom att avkommentera vid pilen i koden. Vad kommer att skrivas ut? Motivera!

7. Redogör för vad som avses med tillstånd och ange så många sätt som möjligt att hantera detta. D.v.s. hur försöker man i imperativ programmering begränsa problemet med tillstånd? 6p

8. Skapa en "frysbar" klass d.v.s. en klass som initialt är muterbar men som senare kan omvandlas till en icke-muterbar. Det är tillåtet att vid behov använda en hjälpklass. Exempel med hjälpklass (annan typ av implementering är tillåten). 8p

```
// Use helper class Creator
Freezable.Creator fc = new Freezable.Creator();
// Possible to set attributes of Freezable using helper, so mutable
fc.setO("aaa");
// ..later ... still mutable
fc.setS("bbb");
//.. now get the immutable object (if not initialized exception or other)
Freezable f = fc.freeze();
// Should not work (exception or other)
fc.setS("ccc");
// Reads should be possible
Object o = f.getO();
String s = f.getS();
// Nope, can't freeze again
Freezable f2 = fc2.freeze();
```

9. Ange för koden nedan vilka rader som kommer att kompilera eller ej. Ange dessutom vilka rader som ger varning? Du måste noggrant motivera dina svar! Ge gärna kodexempel! Number har många subklasser t.ex. Integer och Double. 10p

```
List<?> list1 = new ArrayList<Integer>();
Integer i = list1.get(0);
Object o = list1.get(0);
list1.add(99);
list1.add(o);

List<? extends Number> list2 = new ArrayList<Number>();
o = list2.get(0);
Number n = list2.get(0);
i = list2.get(0);
list2.add(99);

List<? super Integer> list3 = new ArrayList<Number>();
i = list3.get(0);
o = list3.get(0);
list3.add(55);
list3.add(55.0);
```