Hemtentamen i
OOP1
Objektorienterad programmering
Våren 2014

Skriv skriftliga svar på alla frågor om det inte står något annat. Var beredd att munta dina svar.

Lycka till!
Alexander

Hemtentamenför betyg A våren 2014

1 Läs på om *reflections* i java (*java.lang.reflect*). Implementera ett nytt huvudprogram där man kan välja vilket spel som ska spelas och därefter ladda spelet i runtime. Resonera om fördelar och nackdelar med att använda *reflection.* Utgå t.ex. från en spelserver som tillhandahåller javaspel man kan spela online.

2 Skriv en typparametriserad Stack som fungerar som vanligt (a)

```
public class MyStack<E> {
   public Stack();
   public void push(E e);
   public E pop();
   public boolean isEmpty();
}
```

Exempel:
```
MyStack<Number> s = new MyStack<Number>();
s.push(new Integer(3));
s.push(new Float(3.14));
```

Om man skapar en stack av Numbers kan man lägga in både heltal och decimaltal. Lägg till en metod *pushAll* som kan lägga in många element.

```
public void pushAll(Iterable<E> src) {
    for (E e : src)
        push(e);
}
```

Nu går det konstigt nog inte att lägga till en mängd heltal.

```
Iterable<Integer> integers =  new ArrayList<Integer>(Arrays.asList(1, 3, 5, 7, 9));
s.pushAll(integers);  // ERROR pushAll(java.lang.Iterable<java.lang.Number>) cannot be
                              applied to (java.lang.Iterable<java.lang.Integer>)
```

b) Läs på om generics med wildcards och forsök lös problemet genom att ändra på parametertypen i *pushAll*. Läs på och förklara muntligt.

c) Implementera en metod *popAll* ungefär:

```
public void popAll(Collection<E> c) {
      while (! isEmpty())
          c.add( pop() );  }
```

Ändra signaturen så att följande kod kompilerar och kör korrekt:

```
ArrayList<Object> v = new ArrayList<Object>();
s.popAll(v);
```

Förklara ändringen muntligt vid redovisning

d) Vad är fördelarna med att använda typparametrisering? Skriv ner i punktform.

e) Titta i källkoden java/utils/Arrays.java (källkoden finns i src.zip om du laddat hem jdk). Källkoden finns också på nätet t.ex. på docjar.com

google: ”arrays.java java source”

Hur många binärsökningsmetoder (binarysearch etc) finns det i källkoden? Vad är det som skiljer dem åt? Skulle det gå att typparametrisera koden?

3 Ibland är det inte lämpligt med arv. Skriv en klass som ärver från *HashSet* och som håller reda på antal inlagda element någonsin (räknar inte ner när man tar bort element).

```
public class MyHashSet<E> extends HashSet<E> {
    private int count;
    public void Count() { return count; }

    @Override public boolean add(E e) {
        count++;
        return super.add(e);
    }

    @Override public boolean addAll(Collection<? extends E> c) {
        count += c.size();
        return super.addAll(c);
    }
    …
}
```

Koden verkar rimlig men metoden *addAll* fungerar inte som det är tänkt. Vad händer? Varför? Vad vet man om implementationen i HashSet? Finns det något annat sätt man kan lösa det här?

4 Java är ensamt om tvingande undantag. För några år sedan pågick en intensiv debatt om huruvida undantag skulle vara tvingande eller inte. Skumma igenom några av artiklarna genom att söka med Google "exceptions debate", "checked exceptions", "exceptions hejlsberg", ”failure exceptions Gosling”. Reflektera och skriv en egen åsikt i frågan värdigt betyg A. (Det är din reflektion som bedöms inte ditt ställningstagande).

5 Läs på om lambdas i java 8. Skriv en lambdafunktion som returnerar antalet udda tal i en lista med tal.

6 Vilka andra nyheter (inklusive lambdas) finns i java 8. Välj ut minst 5 och högst 10 och beskriv dem kort. Välj i första hand sådana nyheter som (eventuellt med lite kreativt tänkande) skulle kunnat användas i de labbar du skrivit i kursen. Motivera kort varför de skulle kunnat användas eller inte användas.

Reflektera över vad du tycker om respektive nyheter du valt ut. Det är din reflektion som bedöms, inte ditt ställningstagande.