Tentamen 2023-06-01
Objekt-orienterad programmering med Java, 1DL028

Uppsala Universitet
Instutitionen för informationsteknologi
Avdelningen för datalogi

Kursansvarig: Sven-Olof Nyström

31 maj 2023

Läs igenom följande instruktioner noggrant:

- Den första dagen (i alla fall till klockan 15.00) finns jag tillgänglig för frågor via email. Jag kommer förstås att besvara email som vanligt även under resten av tentaperioden.
- Om jag skulle upptäcka någon viktig oklarhet i mina frågor kommer jag att försöka kontakta dig. I så fall skickar jag mail.
- Det är OK att söka information på nätet och i böcker. Det är *inte* OK att be andra människor om hjälp.
- Om bitar av ditt svar har plockats från någon källa, tala om det.
- Svar *måste* motiveras och förklaras. Detta är viktigt. Det blir mycket svårt för mig att ge poäng på svar utan motivering eller program som lika gärna kunde vara tagna från webben.
- I uppgifter som handlar om att skriva program ser jag helst att programmen är debuggade och körklara, men en halvfärdig lösning där tankegången är klar kan ge poäng.
- Några av frågorna handlar om program som du laddar ner här:

  `http://user.it.uu.se/~svenolof/t-230601-graDat`.

  Kontrollera att du kommer åt programmen.
- Vid tentamensperiodens slut (eller när du är klar) laddar du upp svaren och programmen till Studium/Canvas.

Maxpoäng: 40.

Läs igenom hela tentan innan du börjar. Om du fastnar på en uppgift, gå vidare till nästa—uppgifterna är inte nödvändigtvis ordnade i svårighetsgrad. Om en uppgift består av flera deluppgifter och du fastnar på en av dem, gå vidare till nästa deluppgift—det kan hända att efterföljande deluppgifter kan lösas utan det fullständiga svaret på tidigare uppgifter.

1. Referenser (6p)

(a) Anta att vi har en program som implementerar ett patiensspel. En klass som implementerar en hög av kort ser ut så:

```
public class CardPile {

    private List<Card> cards;

    void add(Card c) {
        cards.add(c);
    }

    public List<Card> getCards() {
        return cards;
    }
}
```

Man kan föreställa sig att klassen definierar andra metoder för att modifiera högarna på olika sätt (till exempel ta ett kort från en hög).

Kan kod i andra klasser ändra innehållet i en hög utan att gå via metoder i `CardPile` som är definierade för detta ändamål? Ge motivering.

Ge förslag på bättre sätt att definiera klassen. (Jag kan tänka mig minst två vägar.)

(b) Kan två variabler i ett Java-program referera till samma objekt? Om du tror att det inte är möjligt, tala om varför inte. Om du menar att det är möjligt, förklara hur och ge exempel.

(c) Kan typen av ett objekt ändras? Om du tror att det inte är möjligt, förklara varför inte. Om du menar att det är möjligt, förklara hur och ge exempel.

2. OOP (8p)

(a) Hur ska ett välskrivet objektorienterat program se ut? Utgå från följande punkter.

- Val av klasser.
- Utformning av klassdefinitioner.
- Utformning av enskilda metoder.
- Användning av arv.

(b) Det är ofta lättare att ge exempel på motsatsen, så ge några exempel på typiska situationer när ett program skrivet i Java bryter mot principerna för objektorienterad programmering. Försök att hitta något exempel för varje punkt i föregående delfråga.

3. Solitaire (6p)

Denna fråga och de följande frågorna avser ett program som du laddar ner.

Jag har skrivit programmet så att det kan köras utan inläsning av bildfiler.

Om du vill använda bilder kan du avkommentera kod som läser in bildfilerna (i Card) och kod som ritar upp bilderna (också i Card). Du måste också se

till mappen med bilder ligger på rätt plats och att variablerna `front` och `back` refererar till rätt filer.

Beskriv de olika klasserna och vad de representerar. Vilka klasser representerar korten? Var implementeras spelreglerna? Hur implementeras interaktion med användaren? Hur används arv? Hur används polymorfi?

4. Interaktion (8p)

Frågorna nedan handlar om vad som händer i programmet i olika situationer.

(a) Vad händer (i programmet) när användaren klickar på talongen?

(b) Vad händer när användaren klickar nånstans i spelfönstret?

(c) Hur flyttas ett kort? Vilka metoder anropas? Hur kollar programmet att spelet är enligt reglerna? (Hur kollar programmet att *a*) kortet kan tas och *b*) att kortet kan placeras på den nya platsen?)

(d) Ibland läggs ett kort så att det inte helt täcker det underliggande kortet. I vilka situationer sker detta? Hur avgör programmet när detta ska ske?

(e) Ibland vänds ett kort upp när det flyttas. I vilka situationer sker detta? Hur avgör programmet när detta ska ske?

5. Polymorfi (12p)

Naturligtvis vill vi använda polymorfi i lösningen.

Flytta intelligens, datastrukturer och funktionalitet till klasserna som representerar de olika typerna av korthögar. Använd polymorfi.

Kod som testar typen av ett objekt bör (om möjligt) skrivas om.

(a) vilka metoder måste introduceras?

(b) Beskriv vad som måste ändras

(c) Modifiera programmet så att det använder polymorfi.

LYCKA TILL!

Sven-Olof