LULEÅ TEKNISKA UNIVERSITET

Tentamen i
Objektorienterad programmering och design
Totala antalet uppgifter: 5
Lärare: Håkan Jonsson, 491000, 073-820 1700
Resultatet offentliggörs senast: 2014-09-11.

Kurskod	D0010E
Datum	2014-08-28
Skrivtid	5 tim

Tillåtna hjälpmedel: Inga.

OBS! Lösningar får *inte* baseras på fördefinierade klasser ur t ex Javas standardbibliotek annat än där detta uttryckligen tillåts. Sådana lösningar ger inga poäng. Undantaget `RuntimeException` ska uttryckligen kastas där så är lämpligt.

---

1. Teori

a) Förklara vad orden `abstract`, `final` och `throw` används till i Java. (3p)

b) Klassen `D` i paketet `d` ärver den publika klassen `E` i paketet `e` som ärver den likadeles publika klassen `F` i paketet `f`. I `e` finns även klassen `A`.

Om `E` innehåller en metod `m` deklarerad `protected`, i vilka klasser kan man då använda `m`? (1,5p)

c) Rita UML-diagrammet för deluppgift b. (1,5p)

2. Produkt

Skriv en publik och statisk metod `product` som beräknar produkten av de jämna talen i en heltalsarray med ...

a) ... iteration men utan metodanrop. (3p)

b) ... rekursion/metodanrop men utan iteration. (3p)

Utgå från att produkten av en tom lista är 1.

3. Tärningsfabriken

Din lösning ska i denna uppgift ligga i paketet `dice` och endast bestå av de två klasserna `Dice`, en klass för tärningar, och `DiceFactory`, som kan skapa `Dice`-objekt.

Kod utanför `dice` ska kunna använda typen `Dice` men inte kunna skapa `Dice`-objekt. Behövs en tärning ska det enda sättet vara att anropa `getDice` på en `DiceFactory`.

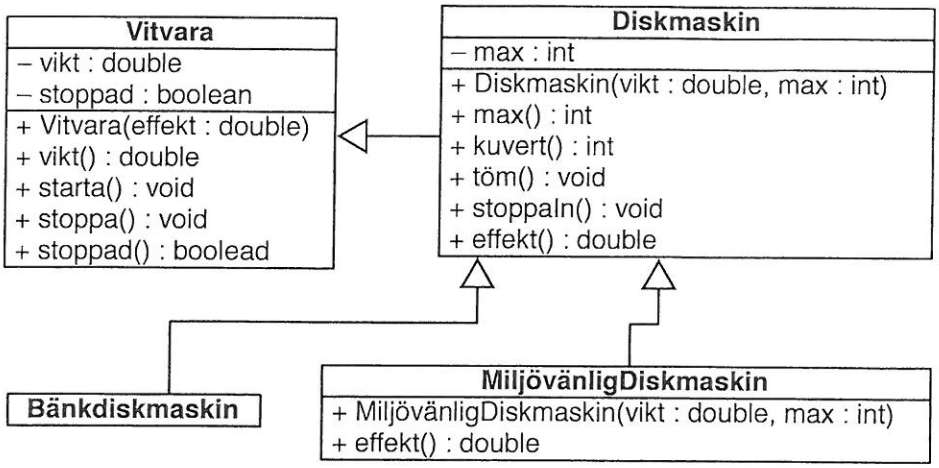
a) Skriv klassen `Dice`. När en tärning skapas anges dess antal sidor n. En tärning har metoden `public int throwDice()` som ger tillbaka ett slumptal som är likformigt fördelat mellan 1 och n. Anrop till `throwDice` motsvarar att man kastar tärningen. Tärningar är, liksom tärningskast, oberoende av varandra.

Du får fritt använda metoden `Math.random()` för att generera likformigt fördelade slumptal x av typen `double` sådana att $0 \leq x < 1$. (3p)

b) Skriv klassen `DiceFactory`. Denna ska endast innehålla metoden `getDice(int n)` som ger tillbaka en n-sidig tärning av typen `Dice`. (2p)

c) Anta att man behöver statistik om skapade tärningar, t ex deras totala antal. Vilka tillägg/ändringar skulle du göra för att åstadkomma detta? (1p)

$n < 1$ ska medföra att undantaget `IllegalArgumentExeption` kastas.



Figur 1: UML-diagram för vitvaror.

4. Vitvaror (6p)

Implementera klasser för "vitvaror" enligt UML-diagrammet. Du får (och **behöver** för full poäng) lägga till nytt i diagrammet men inte ändra befintligt. För klasserna ska följande gälla:

Vitvara Alla vitvaror har en viss effekt (som anges till konstruktorn) och kan startas/stoppas (anrop till dessa metoder sätter variabeln `stoppad` och `stoppad()` returnerar variabeln). Nyskapad är en vitvara stoppad.

Diskmaskin En `Vitvara` som klarar av att diska kuvert[1] med disk (det maximala antalet anges till konstruktorn) och har en effekt på 0.1 kWh/kuvert som diskas. Nyskapad är en diskmaskin tom. Metoden `stoppaIn` ökar antalet kuvert i maskinen med ett (om utrymme finns) medan `töm` gör antalet till 0. Med `max` får man det maximala antal kuvert maskinen kan innehålla och `kuvert` ger aktuellt antal i maskinen. Maskinen ska kunna startas endast om den är stoppad och stoppas endast om den är startad.

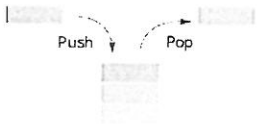
Bänkdiskmaskin En `Diskmaskin` som maximalt kan ta 6 kuvert.

MiljövänligDiskmaskin En diskmaskin som endast har hälften så hög effekt som en `Diskmaskin`.

5. Begränsade stackar

En *stack* kan ses som en lista där element endast kan sättas in och tas bort från den ena änden. Stackmetoden `public void push(Object obj)` sätter in `obj`, medan `public Object pop()` tar ut (bort) och returnerar det *senast* insatta elementet medan `public boolean empty()` avgör om stacken är tom (utan element) eller ej.

En *begränsad* stack har en fix gräns för hur många element den kan innehålla. Denna gräns anges till konstruktorn när en stack skapas.



Figur 2: En stack.

Uppgift: Implementera en begränsad stack med hjälp av

a) ... en array. (3p)

b) ... klassen `LinkedList` och dess metoder som finns listade på nästa sida. (3p)

[1] Ett *kuvert* är en uppsättning tallrikar, bestick, glas mm för en (1) person.

Innehållsförteckning över klassen `LinkedList`[2]:

- `LinkedList()` Constructs an empty list.
- `void addFirst(Object e)` Inserts `e` at the beginning of this list.
- `void addLast(Object e)` Appends `e` to the end of this list.
- `Object getFirst()` Returns the first element in this list.
- `Object getLast()` Returns the last element in this list.
- `Object removeFirst()` Removes and returns the first element from this list.
- `Object removeLast()` Removes and returns the last element from this list.
- `int size()` Returns the number of elements in this list.

[2]OBS! Du ska inte skriva klassen utan bara använda den och metoderna.