Lösningsförslag

Arv

U 1. En referensvariabel har en viss deklarerad typ (klass). Den får då referera till objekt av denna klass eller till subklasser till denna. Därför är satserna 1, 2, 4, och 6 korrekta. Resten är felaktiga.

U 2. `info` får i tur och ordning följande värden: "Lisa Svensson", "Lisa Svensson, D", "Kalle Karlsson, C" och "Per Holm, Computer Science".

Förklaring till den andra tilldelningen: `p` har deklarerad typ `Person`, men refererar här under exekvering till ett objekt av typen `Student`. Det blir därför metoden `toString()` i klassen `Student` som exekveras.

Förklaring till fjärde tilldelningen: `p` refererar här till ett objekt av typen `FacultyMember`.

U 3. Det beror på vilken typ av objekt `ref` refererar till under exekveringen. Om `ref` refererar till ett objekt av subklassen, så blir det metoden `p` där som exekveras, annars blir det `p` i klassen `C`.

U 4. Alternativ 1 och 4 är korrekta. Korrekt antal parametrar skickas med till konstruktorn i bägge fallen. Eftersom `Shape` är superklass till `Square` får `s` referera till `Square`-objekt. I alternativ 2 stämmer inte antal parametrar. Alternativ 3 är fel eftersom man inte kan skapa objekt av abstrakta klasser.

U 5. Vektorelementet `theShapes[i]` har den deklarerade typen `Shape`. Vid kompileringen söks därför efter en metod `draw()` i klassen `Shape`. Men någon sådan metod finns inte. Lösningen är att deklarera `draw` som en abstrakt metod i klassen `Shape`:

```
public abstract void draw();
```

Parameteröverföring

U 6. 2 skrivs ut. Förklaring: När en parameter förs över till en metod kopieras den. Inuti metoden `incr` är det kopian som ökas med 1.

U 7. "Lisa" skrivs ut. Förklaringen är densamma som i föregående uppgift. Metoden har en referensvariabel som parameter. När referensen `p` förs över som parameter, kopieras den. Vi kan kalla kopian `pcopy`. Eftersom den är en kopia på `p` så refererar den först till samma objekt som `p` (Lisa). Inuti metoden får sedan referensen `pcopy` ett nytt värde. Den refererar nu till ett annat objekt. Men `p` ändras inte.

U 8. Som ovan, `p` kopieras och vi kan igen kalla kopian `pcopy`. `pcopy` refererar till personobjektet som har namnet "Lisa". I metoden anropas sedan metoden `setName` för den person som `pcopy` refererar till. Det resulterar i att denna person får nytt namn, "Kalle". Därför kommer "Kalle" att skrivas ut.

U 9. 10 skrivs ut. I Java är vektorer objekt. Vi kan alltså uppfatta `a` som en referens till det objekt som representerar vektorn. Förklaringen är sedan densamma som i föregående uppgift: När vi anropar metoden är det egentligen en kopia av referensen som används i metoden. Men den refererar till samma vektorobjekt.

U 10. Vektorn `nbrs1` innehåller 10, 20, 30, 40, 50 och `nbrs2` innehåller 50, 40, 30, 20, 10.

I bägge metoderna avser man att vända på elementen i vektorn a så att de kommer i omvänd ordning. Men metoden `r1` innehåller ett fel så att vektorn förblir oförändrad. När referensen `nbrs1` vid anropet av `r1` förs över som parameter, kopieras den till `a`. `a` refererar alltså först till samma vektor som `nbrs1`. Inuti metoden får sedan `a` ett nytt värde och refererar nu till `temp`-vektorn. Men innehållet i `nbrs1` ändras inte.

När `r2` anropas kopieras referensen till vektorn `nbrs2` till `a`. När vi inuti metoden ändrar i vektorn `a` är det i själva verket vektorn `nbrs2` som ändras.

Statiska metoder och attribut

U 11.

a) Statiska attribut och metoder är knutna till en klass, inte till ett objekt. De finns bara i en upplaga per klass. I detta fall har man valt att deklarera attributet `static` eftersom (vi antar att) alla personer har samma lagstadgade pensionsålder.
I statiska metoder kan man inte komma åt icke-statiska attribut eller metoder. I vanliga (icke-statiska) metoder kan man komma åt både vanliga attribut och metoder och statiska attribut och metoder. Metoderna `setSeniorAge` och `getSeniorAge` kan alltså implementeras som antingen vanliga metoder eller (som här) statiska metoder. Man föredrar dock att deklarera metoder som statiska när så är möjligt. Detta beror på att vanliga attribut och metoder har en (implicit) referens till det objekt de är knutna till. Detta är onödigt i de fall vi bara hanterar statiska attribut (eller inga attribut alls).

b) I det första fallet använder man sig av klassnamnet (`Person`) för att referera till det statiska attributet. I det andra fallet använder man sig av ett objekt av klassen för att komma åt samma attribut. Båda är korrekta men man brukar föredra att använda klassnamnet.

c) Det statiska attributet ändrar värde till 65 genom satsen `p.setSeniorAge(65)`. Ett statiskt attribut är gemensamt för alla objekt av klassen `Person`. Därför skrivs 65 ut.