

1. a) PRIVATE:
åtkomst endast inom den egna klassen.
private är den högsta skyddsnivån

b) PROTECTED:
åtkomst inom den egna klassen, i klasser
i det egna paketet och eventuella sub-
klasser i andra paket. (bra vid arv)

c) PUBLIC - åtkomst överallt i programmet.
public är den lägsta skyddsnivån.

Om man inte anger någon modifierare
blir det automatiskt package, Bra!
dvs åtkomst/synlighet endast i klasser
inom det aktuella paketet.

3

Mycket bra svar
på samtliga frågor! 48/50
Re VG

②



Namn

Personnummer

Personnummer

Blad nr

Uppgift nr

2. static markerar klassstillhörigheten (& inte instanstillhörighet)

När man säger static framför en variabel blir det en klassvariabel.

Klassvariabler är gemensamma för alla objekt som tillhör klassen. Det finns bara en av den och värdet är detsamma för alla objekt som tillhör klassen.

(En metod som markeras static blir en klassmetod. En klassmetod hanterar inga individuella objekt utan använder bara klassvariabler.)

(Dvs alla instanser som skapats av en klass som markerats som static kommer att dela på värdet.)

(En klassmetod anropas enligt följande:
klassnamn.metodnamn(argument))

heltalsvariabler får typen Int dvs
Int står för heltal.

2

3



Namn

Personnummer

Blad nr

Uppgift nr

3). $x = 9$ $i = 9$

initiering

en forloop består av (~~iteration~~, villkor och förändring)
forloopen säger:

så länge som i är större eller lika med 0
så i minskas med 1 ($i--$)

($i--$ betyder att i ska användas först innan
det minskas med ett.)

Utskriften kommer att bli: $i * x$: resultat
för varje utskrift kommer man ^{it}hoppa en rad
(`System.out.println()`)

$i = 9$ $x = 9$

$i = 8$ $x = 9$

$i = 7$ $x = 9$

VARV 1: $9 * 9$: _81 VARV 2: $8 * 9$: _72 VARV 3: $7 * 9$: _63

$i = 6$ $x = 9$

$i = 5$ $x = 9$

$i = 4$ $x = 9$

VARV 4: $6 * 9$: _54 VARV 5: $5 * 9$: _45 VARV 6: $4 * 9$: _36

$i = 3$ $x = 9$

$i = 2$ $x = 9$

$i = 1$ $x = 9$

VARV 7: $3 * 9$: _27 VARV 8: $2 * 9$: _18 VARV 9: $1 * 9$: _9

$i = 0$ $x = 9$

VARV 10: $0 * 9$: _0

Nästa varv blir $i = -1$ och då gäller inte
villkoret längre.

PÅ NÄSTA BLAD FÖLJER UTSKRIFTEN

VAR VÄNLIG VÄND TILL NÄSTA
BLAD →

6

4



Ämne

Personnummer

Blad nr

Öppettid

FORTSÄTTNING FRÅGA ③
utsnitten kommer att bli:

$$9 * 9: _ 81$$

$$8 * 9: _ 72$$

$$7 * 9: _ 63$$

$$6 * 9: _ 54$$

$$5 * 9: _ 45$$

$$4 * 9: _ 36$$

$$3 * 9: _ 27$$

$$2 * 9: _ 18$$

$$1 * 9: _ 9$$

$$0 * 9: _ 0$$

(Dvs 9:ans tabell)

— står för tabb = \t
alltså att det blir ett mellanrum mellan : och resultat.

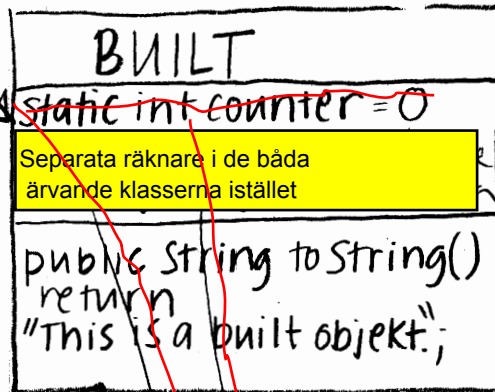
5



4)

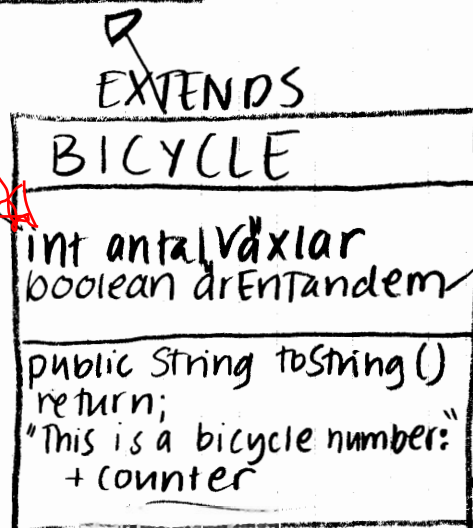
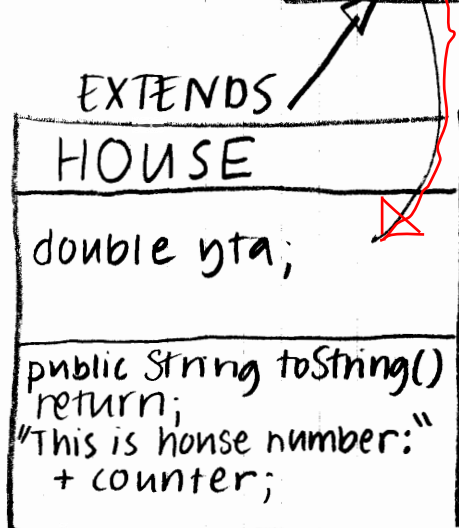
public class Built {

klassvariabel,
alla objekt som
skapas av klassen
kommer att få
värdet 0.



DÄ JAG INTE ANÖETT
NÅGON MODIFIERARE
(SKYDDSNIVÅ) PÅ DE OLIKA
VARIABLERNA FÅR DE
AUTOMATISKT SKYDDSNIVÅN
PACKAGE -

dvs - åtkomst endast i klasser
i det egna paketet.



6

om boolean
inte tilldelas
något värde
blir den falsk
vilket i detta fall
är mest troligt då
de flesta bicycles
(cyklar) inte är
tandemcyklar.

KONSTRUKTOR:
public Built(){
++counter;
}

För varje objekt som skapas av
klassen kommer counter att
ökas med ett och på så sätt
får varje objekt ett "eget" nummer.

Alla variabler förutom counter är instansvariabler

int = står för heltal, tal utan decimaler

double = flyttal, tal m. decimaler (5 siffrors noggrannhet)

house och bicycle ärver built.

I javahod skulle det se ut:

```
public class House extends Built {  
public class Bicycle extends Built {
```

Extends = ärva
en klass utav
en annan.
Man kan bara
ärva utav
en klass.

6



Personnummer

Blad nr

Uppgift nr

5. Överskuggning är när en metod i superklassen omdefinieras för att passa subklassen. Subklassens metod överskuggar då metoden i superklassen.

I föregående uppgitt omdefinierades superklassens metod just för att den skulle passa för de två subklasserna.

(När en metod överskuggar en annan är det alltså för att metoden ska passa för en viss klass.)

Dynamisk bindning är när rätt metod automatiskt körs vid exekvering.

Javainterpretatorn börjar längst ner i arvshierarkin och söker sedan uppåt till den hittar en metod som matchar de angivna parametrarna.

(Vid dynamisk bindning har metoderna samma namn & signatur men de måste ligga i olika klasser.)

(Även superklassens metod är en överskuggning av metoden `String toString()`)

`toString()` i klassen `Built` är en överskuggning av den `toString` som finns i klassen `Object`

Mycket bra svar!

4

7



Personnummer

Blad nr

Uppgift nr

6.) selektion är när vi i våra program ställs för olika vägval.

Vi behöver sätt att styra våra vägval. Detta kan vi göra genom att använda oss av if-else & switchsater.

Switchsater passar bäst när vi har många vägval (från 3)

en switchsats är uppbyggd av olika case, vilka tilldelas varsitt värde. Finns inte värdet med bland de olika case - alternativen går man vidare till default - alternativet om det finns något. if, else kan se ut på följande vis:

tex: `if (x > 5) {`

 så ska följande hända

else...

(etc.)

6

Iteration betyder upprepning och i java görs iteration med programslingsor (loopar) hur många gånger en kod ska itereras bestäms av ett villkor.

VÄND TILL NÄSTA BLAD →



Namn

Personnummer

Blad nr

Utgifts nr

FORTSÄTTNING FRÅGA (6)

villkor ställs med följande konstruktioner:

while - villkoret testas först

do-while - fungerar som while men villkoret testas sist.

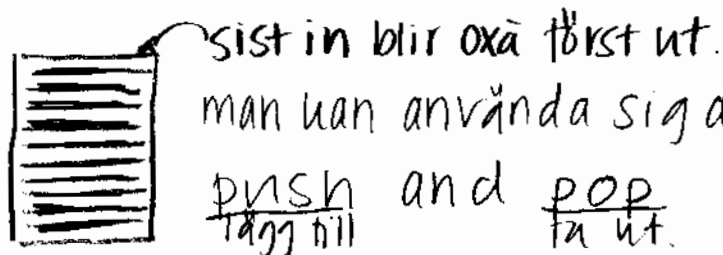
for-loop - passar bäst när man vet hur många gånger koden ska itereras.
for (initiering; villkor; förändring)

initiering

7) En stack fungerar som en tallnushög i en matbespishning.

Dvs. sist in först ut = LIFO (Last in first out)

Det objekt som lagts till sist i stacken är också det man tar ut först.



Vi använde oss av stacken under våra laborationer när vi skulle konstruera ett palindrom.

(Dvs ett ord som blir samma sak framlänges som baklänges)

Vi behövde då ett sätt att jämföra ett ord bokstav för bokstav (framlänges & baklänges)

Därför behövde vi också använda oss av en kö.

Queuen kö fungerar som en stack fast tvärtom.

FIFO - first in first out →

5	4	3	2	1
---	---	---	---	---

 de objekt man stoppat in först är också det objekt som man plockar ut först.

Genom att använda oss av kö & stack kunde vi jämföra en textsträng, framåt & bakåt.

Dvs om den blev samma sak bakåt som framåt dvs om det var ett palindrom eller inte.

10



Namn

Personnummer

Blad nr

Uppgift nr

8). En array är ett fält/benhållare för flera objekt av samma typ. Fälten är numrerade och indexeringen börjar alltid vid noll.

De olika minnesplatserna kan tilldelas egna värden oberoende av varandra.

En array måste alltid initieras.

tex : `int[] heltal = new int[5]` arrayen tilldelas värden
eller: `int[] heltal = {5, 8, 10, 9, 7}` ←

om man tex skriver:

`int x = heltal[2]` så menar man
talet på position 2 som i detta
fallet är 10.

Om ett tal finns inom `[1]` menar man alltid
på position 1. (inte 1 som tal i sig).

4

En konstruktor känns igen enligt följande:

17

1. är alltid public (public)
2. har ingen returtyp
3. heter alltid samma sak som klassen.



Namn

Personnummer

Blad nr

Uppgift nr

9.) En konstruktor är en slags initieringsmetod som anropas automatiskt varje gång ett nytt objekt skapas av klassen.
En överlagrad konstruktor =

Det kan finnas flera konstruktörer med samma namn i en klass men de måste ha olika parametrar (dvs. argumentlistor)

→ antingen olika antal parametrar
→ eller olika typer på parametrarna.

I uppgitt två ITKPerson och dess subklasser förstod jag konstruktorernas funktion och hur argument sänds med.

I basklassen använde jag mig av konstruktörer på följande sätt:

```
public ITKPerson(String namn) {  
    this.namn = namn;  
}  
och:
```

```
public ITKPerson(String namn, String ttnNummer)  
    this.namn = namn;  
    this.ttnNummer = ttnNummer;  
}
```

6