2)

```
public class Uppg2 {
	public static void main(String[] args){
		int a=10;
		for(int i=0; i<a; i++){
			boolean b = true;
			while(b){
				int e=0;
				do {
					Integer c = null;
					if(c==null){
						int d;
						d = a++;
					}
				} while(e!=a);
			}
		}
	}
}
```

3)

```
public class Uppg3 {
	//a
	public static void printaArray(int[] numberArray){
		for(int i = 0; i<numberArray.length; i++){
			System.out.print(numberArray[i]);
		}
	}
	//b
	public static int[] kopieraArray(int[] numberArray){
		int[] tempArray = new int[numberArray.length];
		for(int i = 0; i<numberArray.length; i++){
			tempArray[i] = numberArray[i];
		}
		return tempArray;
	}
	//c
	public static void setZeros(int[] numberArray){
		for(int i=0; i<numberArray.length; i++){
			numberArray[i]=0;
		}
	}
	//main-metod som testar metoderna
```

```
	public static void main(String[] args){
		System.out.println("a)");
		int[] numberArray = {1,2,3,4,5,6,7,8,9};
		printaArray(numberArray);

		System.out.println("\nb)");
		int[] newArray = kopieraArray(numberArray);
		printaArray(newArray);

		System.out.println("\nc)");
		setZeros(numberArray);
		printaArray(numberArray);
	}
}
```

4)

```
public class Monster {
	private int nbrOfArms;
	private int nbrOfLegs;
	private int nbrOfEyes;
	private int nbrOfNoses;

	//Skriv konstruktorer här:

	//a
	public Monster(int arms){
		nbrOfArms = arms;
	}

	//b
	public Monster(int legs, int eyes){
		nbrOfLegs = legs;
		nbrOfEyes = eyes;
	}

	//c
	public Monster(int arms, int legs, int eyes, int noses){
		nbrOfArms = arms;
		nbrOfLegs = legs;
		nbrOfEyes = eyes;
		nbrOfNoses = noses;
	}
```

```
	//d
	/*
	 * Om inga andra konstruktorer hade deklarerats hade man
	 * kunnat anropa defaultkonstruktorn utan problem.
	 * Nu när ni har skrivit andra konstruktorer fungerar
	 * detta dock inte längre och ni måste även deklarera en
	 * defaultkonstruktor.
	 *
	 * Prova först att anropa new Monster() och se vad som händer.
	 *
	 * Skriv därefter en defaultkonstruktor, t.ex såhär:
	 */
	public Monster(){
		nbrOfArms = 0;
		nbrOfLegs = 0;
		nbrOfEyes = 0;
		nbrOfNoses = 0;
	}

	public static void main(String[] args){
		/*Testa att anropa de olika konstruktorerna här och se om
		 * du får vad du trodde på olika värdena.
		 * T.ex:
		 */
		Monster monster1 = new Monster();
		System.out.println(monster1.nbrOfArms);
	}
}
```

5)

```
public class Uppg5 {
	public static int rekursivAddition(int bigger, int smaller){
		if(bigger<smaller){
			int temp;
			temp=bigger;
			bigger=smaller;
			smaller=temp;
			return rekursivAddition(bigger, smaller);
		}
		else if(smaller>0){
			bigger++;
			smaller--;
			return rekursivAddition(bigger, smaller);
```

```
            }
            else{
                  return bigger;
            }
      }
      public static void main(String[] args){
            //Test
            System.out.println(rekursivAddition(4, 3));
            System.out.println(rekursivAddition(3, 4));
            System.out.println(rekursivAddition(0, 0));
            System.out.println(rekursivAddition(0, 8));
            System.out.println(rekursivAddition(8, 0));
            System.out.println(rekursivAddition(8, 8));
      }
}
```

överkurs)

```
/*
 * Obs. man kan lösa denna på flera sätt. Detta är ett alternativ.
 */
public class Uppg5 {
      public static int rekursivAddition(int higher, int lower){
            /*Check that higher has a higher number than lower
             *In this case higher means a bigger int-value without
             *concerning plus and minus
             *(Example: -3 is "higher" than 2)
             */
            if(Math.abs(higher)<Math.abs(lower)){
                  int temp;
                  temp=higher;
                  higher=lower;
                  lower=temp;
                  return rekursivAddition(higher, lower);
            }
            else if(lower>0){
                  higher++;
                  lower--;
                  return rekursivAddition(higher, lower);
            }

            else if(lower<0){
                  higher--;
                  lower++;
```

```
                return rekursivAddition(higher, lower);
            }
            else {
                return higher;
            }
        }

        public static void main(String[] args){
            //Test
            System.out.println(rekursivAddition(4, 3));
            System.out.println(rekursivAddition(3, 4));
            System.out.println(rekursivAddition(0, 0));
            System.out.println(rekursivAddition(0, 8));
            System.out.println(rekursivAddition(8, 0));
            System.out.println(rekursivAddition(8, 8));
            System.out.println(rekursivAddition(0, 0));
            System.out.println(rekursivAddition(8, -8));
            System.out.println(rekursivAddition(-8, 8));
            System.out.println(rekursivAddition(8, -8));
            System.out.println(rekursivAddition(-8, -8));
            System.out.println(rekursivAddition(-8, -3));
            System.out.println(rekursivAddition(-3, -8));
            System.out.println(rekursivAddition(3, -8));
            System.out.println(rekursivAddition(-8, 3));
            System.out.println(rekursivAddition(-3, 8));
            System.out.println(rekursivAddition(8, -3));
            System.out.println(rekursivAddition(-8, -2));
        }
}
```

Kämpa på nu, och glöm inte att varje gång ni kör fast så håller ni på att lära er något nytt! Det är alltså bra att köra fast. När man kört fast så testar och felsöker man koden på olika sätt, läser boken, kollar föreläsningsanteckningar, googlar och/eller frågar sina kompisar. Då hittar man svaret till slut!

Den eller de som på tisdag kan presentera en egen smart/snygg lösning av den rekursiva överkurs-uppgiften kommer att få en överraskning:)
Kör hårt!
//Anton