1. Beskriv (kortfattat och översiktligt) vad Streams är för något och vad det kan vara bra för.

Stream är en sekvens av objekt som har olika metoder vilket kan ses som ett löpande band där varje objekt/metod är som en station som gör något med till exempel en lista så som filtrera den eller ändra på värden som finns i den.

Streams är bra för att det blir mindre och mer läsbar kod en om man gör på det mer traditionella sättet. Exempel så behövs bara 1 lista i streams när man ska filtrerar medan utan streams behövs det 2 listor, minst 1 for-loop, minst 1 if-sats och en separat forEach-loop för att skriva ut resultatet.

1. Skriv koden till ett eget funktionellt interface samt ge ett exempel på en    
   implementation av ditt interface på lambda-form.
2. *@FunctionalInterface  
   interface* Function {  
    *int* add(*int* x);  
   }  
   *@FunctionalInterface  
   interface* GenericFunction<T, U, M,R> {  
    R apply(T t, U u, M m);  
   }  
   *public class* Main {  
     
    *public static void* main(String[] args) {  
     
    Function multi = x -> x \* 2;  
    GenericFunction<Integer,Integer,Integer,Integer> genericFunction = (t,u,m) -> t \* u \* m;  
      
    }  
   }

1. Beskriv kortfattat och översiktligt varför programmerar använder design patterns?

Design patterns är programmeringsteknik som hjälper programmerare att lösa återkommande problem, skapar en bra struktur på koden och kollegor kan lättare förstå vad det är man vill koden ska göra. Gör koden mer flexibel till förändringar med.

1. Nämn två olika design pattern och förklara hur *ett* av dessa fungerar samt vad det kan vara bra för.

Observer Pattern

Command pattern

Command pattern är som det låter ett kommando som skickas till en eller flera mottagare. Tänk att man har en kontroll som kan stänga av och på lampor eller vad som helst som har en av och på knapp (finns fler saker som den kan göra detta pattern) när man trycker **på** så kommer det skickas ut ett ”kommando” till mottagarna att de ska starta och lika dant med **av** knappen.

Hur det funkar är att det ska finnas en **”Invoker” klass** som har en **”Command” interface**

**”Command” interfacet** blir implementerat av en **”konkret Command” klass** som sedan har en **”Recevier” klass**

**Invoker** klassen ska sätta kommandot, tänk detta som en fjärrkontroll.

**Command** interfacet har två metoder som **execute** kommandot och en som **Undo** kommandot

**Command klassen** har samma metoder då den implementerar från **Command interfacet**, plus att varje metod kallar på **Recevier action metoden** och det är i den metoden som ska göra något beroende på vad för kommando den har fått.

Klienten är viktigt men den är inte hjärtat av själva **command patternet**



1. Beskriv (kortfattat och översiktligt) vad reguljära uttryck / regex är för något?

Det är ett sätt att kunna matcha olika mönster i strängar som ett telefonnummer börjar på korrekta siffror eller om en fil slutar på .pdf.

Kan användas när vi vill hindra användaren att skicka iväg en inkorrekt mail-adress

1. Ge exempel på ett reguljärt uttryck och beskriv vad ditt uttryck matchar på.
2. ([aeiouy]\\w\*){2,}

Hittar alla ord som har mer en 2 vokaler