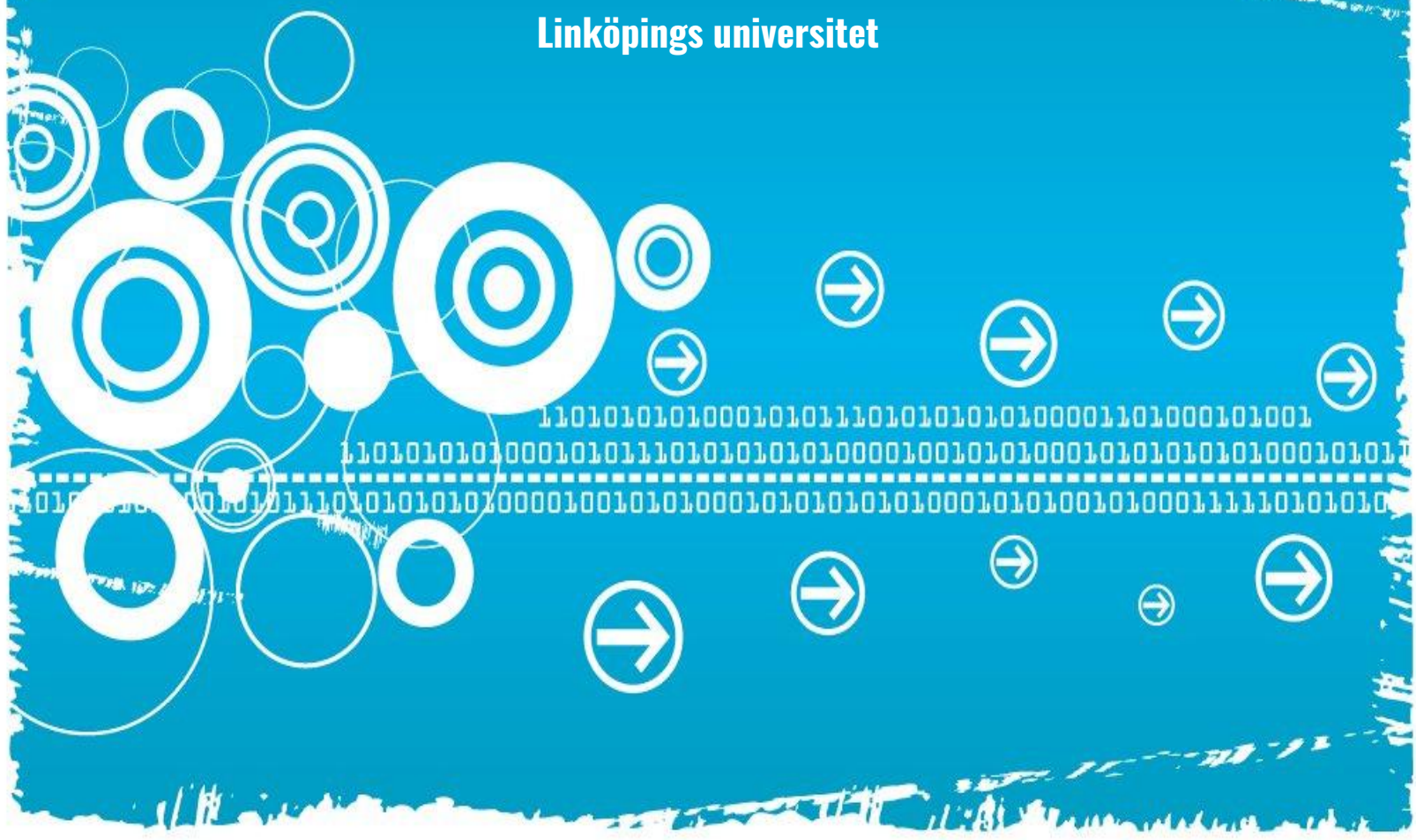
TDDC77 - Objektorienterad programmering

Laboration 3

Institutionen för datavetenskap
Linköpings universitet



3.1 Calculator*

Nu är det dags att implementera och använda din första klass. Klassen heter `Calculator` och presenterar en kalkylator som innehåller följande huvudmetoder:

- `add(int op1, int op 2)`
- `minus(int op1, int op2)`
- `multiply(int op1, int op2)`
- `divide(int op1, int op2)`

Returtypen ska bestämmas så att den täcker alla möjliga värden. Klassen innehåller också ett minne som kan uppdateras och läsas. Minnet implementeras som ett fält av typen ArrayList. Följande metoder ska finnas för att uppdatera och läsa minnet:

- `addToMemory`
    - Den här metoden tar emot ett värde och lägger till det till minnet.
- `getMemory`
    - Den här metoden returnerar alla värden som ligger i minnet som en vanlig array. Den här metoden får inte göra någon utskrift.

Antal parametrar, parametertyperna och returtyperna bestäms så att alla möjliga scenarion täcks. Alla huvudmetoder ska använda `addToMemory` för att lagra resultatet i minnet. Med andra ord varje resultat som genereras och visas av huvudmetoderna ska också lagras i minnet.

Det ska finnas en `Main` klass som är ansvarig för att instansiera och använda `Calculator`. `Main` är ansvarig för att läsa in två heltal, tillåta användaren välja vilken huvudmetod ska användas, anropa den valda huvudmetoden och sedan skriva ut resultatet för användaren. Innan varje inmatning och utmatning ska finnas en utskrift för att instruera användaren.

`switch` ska användas i `main` metoden för att bestämma vilken huvudmetod ska anropas.

Exempel

```
Choose an operation(+, -, *, /): +
Enter an integer: 12
Enter another integer: 24
Result: 36
========================
Enter an integer:
```

Så fort användaren ser resultatet ska programmet upprepa hel processen. Det ska finnas en delimiter som visas innan nästa upprepning. Om användaren matar in x som operatorn då ska alla resultat som ligger i minnet visas för användaren innan programmet avslutas.

För att visa minnet använd gärna följande format:
`[ value_01, value_02, value_03, …, value_n ]`

Exempel

```
Choose an operation(+, -, *, /): +
Enter an integer: 12
Enter another integer: 24
Result: 36
========================
Choose an operation(+, -, *, /): *
Enter an integer: 1
Enter another integer: 56
Result: 56
========================
Choose an operation(+, -, *, /): x
[36, 56]
```

Alla klasser ska ligga i följande paket: `se.liu.ida.tddc77.lab3`

Nu ska du utöka programmet genom att lägga till en ny unär operation som finns i `java.lang.Math`. Du får bestämma vilken symbol som helst för den nya operationen. Programmet ska läsa in bara en operand i fall den nya operationen har valts av användaren.

Exempel

```
Choose an operation(+, -, *, /, []): []
Enter an integer: 12.6
Result: 12
========================
Choose an operation(+, -, *, /):
```

Noteringar:

- Main ska bara instantiera `Caluculator` bara en gång.
- `Calculator` får INTE läsa in eller skriva ut data.
- `Main` är ansvarig för kommunikation med användaren.
- `Main` bestämmer hur data ska presenteras för användaren.
- `Main` ska använda `java.lang.Math` direkt.
- Det är inte tillåtet att använda `for` alls men däremot är `do-while` och `while` tillåtna.