

- i** Einu leyfilegu hjálpargögnin eru **eitt A4 blað (handskrifað báðum megin)**, sem nemandinn hefur búið til.

Ef þú **sérð ekki** hluta prófsins, t.d. hlekki eða myndir, gefðu þig fram við **prófvörð**

Ef **galli á prófinu kemur í ljós**, verður tekið tillit til þess við yfirferð prófsins.

Ef nemandi er óviss um hvort skilningur hans á forritunarspurningu er réttur, getur hann tilgreint hver sá skilningur er í svarreit í opnum spurningum eða sent póst til kennara eftir prófið

Farið verður **nafnlaust** yfir prófið þannig að þið skuluð ekki skrifa nafnið ykkar í lausnina.

Í viðhengi (resources) eru eftirfarandi

Snara.is

PrintWriter

File

Scanner

Math

- i** Svárið eftirfarandi krossaspurningum með því að merkja við eitt svar í hverri spurningu. Ef ykkur finnst fleiri en eitt svar koma til greina skuluð þið velja svárið sem ykkur finnst „réttast“. Ekki er dregið niður fyrir rangt svar.


- 1 Skoðið eftirfarandi klasa Account sem er líkan af bankareikningi. Klasinn hefur tvær tilviksbreytur balance (innistæða reiknings) og vexti (interestRate).

Hér er byrjunin á klasanum

```
public class Account {  
    private int balance;  
    private int interestRate;  
    public Account (int balance, int interestRate) {  
        this.balance = balance;  
        this.interestRate = interestRate;  
    }  
  
    public int getBalance() {  
        return balance;  
    }  
  
    // fleiri aðferðir klasans  
}
```

Hvað er satt um getBalance() aðferðina

Veldu eitt af eftirtöldu:

- ☐ leggur inn á reikninginn
- ☐ skilar parameter í smiðnum fyrir innistæðu reiknings
- ☒ skilar innistæðu reiknings 
- ☐ skilar heildarinnistæðu allra reikninga
- ☐ skilar vöxtum reiknings

Maximum marks: 6


- 2 Skoðið eftirfarandi klasa Account sem er líkan af bankareikningi. Klasinn hefur tvær tilviksbreytur balance (innistæða reiknings) og vexti (interestRate).

Hér er byrjunin á klasanum

```
public class Account {  
    private int balance;  
    private int interestRate;  
  
    public Account (int balance, int interestRate) {  
        this.balance = balance;  
        this.interestRate = interestRate;  
    }  
  
    public int getBalance() {  
        return balance;  
    }  
  
    // fleiri aðferðir klasans  
}
```

Hvaða fullyrðing um smíð Account klasans er rétt?

Veldu eitt af eftirtöldu

- ☒ Smíðurinn notar this til að setja tilviksbreyturnar 
- ☐ Skilgreining smíðsins er röng og ætti að vera public Account Account (int balance, int interestRate)
Útskýring: Þó svo að smíðurinn skili Account hlut hefur skilgreining ekki tagið á skilagildinu
- ☐ Það vantar return setningu neðst í smíðinn
Útskýring: Smíður skilar Account hlut og ekki þarf að skila sérstaklega
- ☐ Í staðinn fyrir setningarnar tvær með this mætti nota super (balance, interestRate);
Útskýring: Account erfir aðeins frá Object sem hefur smíð án parameter
- ☐ Smíðurinn setur tilviksbreyturnar með sjálfgefnum gildum
Útskýring: Smíður setur tilviksbreytur með parameter gildunum

Maximum marks: 6

3 Hvað prentast út?

```
public static void main(String[] args) {  
    int[] fylki = new int[] { 3, 2, 9, 1, 10, 7, 11 };  
    for (int i = 0; i < fylki.length/2; i++) {  
        int t = fylki[(fylki.length - 1) - i];  
        fylki[(fylki.length - 1) - i] = fylki[i];  
        fylki[i] = t;  
    }  
    System.out.println(Arrays.toString(fylki));  
}
```

Veldu eitt af eftirtöldu

- ☒ [11, 7, 10, 1, 9, 2, 3]
- ☐ [3, 2, 9, 1, 10, 7, 11]
- ☐ [11, 7, 10, 1, 9, 2, 3, 0]
- ☐ [11, 7, 10, 1, 9, 2]
- ☐ [3, 7, 10, 1, 9, 2, 11]




Maximum marks: 6

4 Skoðaðu eftirfarandi forrit

```
public class Test {  
    private int x;  
    Test(int x) {  
        System.out.println("Test");  
    }  
    public static void main(String[] args) {  
        Test test = null;  
        System.out.println(test.x);  
    }  
}
```

Hvert eftirfarandi á best við?

- ☐ Þegar forritið er þýtt verður til villan "variable test has not been initialized"
- ☐ Forritið býr til NullPointerException þegar það er keyrt 
- ☐ Forritið prentar út "Test" þegar það er keyrt
- ☐ Forritið prentar út "null" þegar það er keyrt
- ☐ Þegar forritið er þýtt kemur villa því Test klasinn hefur ekki sjálfgefinn smíð

Maximum marks: 6

Lausn: Breytan test inniheldur null. Þegar test.x er keyrt er reynt að ná í tilviksbreytu x í null hlut sem framkallar NullPointerException

- 5 Skoðið eftirfarandi þrjá klasa A, B og C. main fallið í C býr til tvo hluti af klösunum A og B. Klasinn A erfir frá B. Klasarnir A og B hafa aðferðirnar toString() - Hvað er prentað út þegar main fallið í C er keyrt?

```
public class C {  
    public static void main(String[] args) {  
        Object[] fylki = { new A(), new B() };  
        System.out.print(fylki[0]);  
        System.out.print(fylki[1]);  
    }  
}
```

```
public class B {  
    public String toString() {  
        return "B";  
    }  
}
```

```
public class A extends B {  
    public String toString() {  
        return "A";  
    }  
}
```

Veldu eitt af eftirtöldu

Útskýring: toString í klasanum A yfirskrifar (overrides) toString í klasanum B þannig að fylki[0].toString() skilar "A". fylki[1].toString() skilar "B".

- ☒ AB
- ☐ ABB
- ☐ BA
- ☐ AA
- ☐ BAB



Maximum marks: 6

- 6 Klasinn **ProfException** erfir (extends) frá **Exception** og hefur eina viðbótar tilviksbreytu af taginu boolean sem heitir **success**. (Ekki sýndur hér).

Klasinn **Lestur** hér fyrir neðan hefur eina aðferð **lesa** sem tekur inn **Scanner** sem parameter og skilar heiltölu.

main fallið í klasanum **Lestur** kallar á aðferðina **lesa** og grípur viðeigandi exception og bregst við því. (try - catch).

Fyllið inn í eyðurnar hér á eftir

```
import java.util.InputMismatchException;
import java.util.Scanner;
```

```
public class Lestur {
```

```
    public static  (int) lesa (  (Scanner inntak) )
```

```
     (throws, throw)  (ProfException) {
```

```
        int tala;
        try {
            tala = inntak.nextInt();
        }
        catch (InputMismatchException e) {
            throw (new ProfException(false, e));
        }
        return tala;
    }
}

// main aðferð- ekki sýnd
}
```

Útskýring:

public static int lesa (Scanner inntak) throws ProfException

Maximum marks: 8

- i** Í forritunardæmum er gefinn java ritill til að forrita í. Opni ritillinn er með setninganúmerum og sýnir java highlights með lit. Ritillinn hjálpar ykkur með réttan inndrátt á línum **ef þið notið slaufusviga**. Þegar gefin er byrjun á forriti eða beinagrind getið þið afritað hana í opna ritilinn. Í opna ritlinum geturðu ekki prófað forritið.

Í code compile spurningum er gefin byrjun á forriti. Sá ritill er ekki með línunúmer og þú þarft að sjá um inndráttinn. Munið að ýta á Test code til að prófa forritið

Ef gefin eru leiðbeinandi línukomment fylgið leiðbeiningum og eyðið þeim ekki.

Notið **bestu venjur** við forritun. Notið aðeins þá klasa sem farið hefur verið yfir í námsefninu.

Þið þurfið **ekki að lýsa forritinu** með því að bæta við haus eða javaDoc nema sé beðið um það sérstaklega. Þið þurfið **ekki að bæta við import** setningum efst í forritið ykkar

Forritið **static** aðferðina **reiknaFormula** sem tekur inn þrjá parametra **x**, **y**, **z** sem hver er **heiltölufylki** af óþekkttri stærð en þau eru jafnlöng. **reiknaFormula** skilar (e. returns) engu.

Aðferðin reiknar út **z** samkvæmt eftirfarandi. z_i er samsvarandi $z[i]$ o.s.frv. fyrir **x** og **y**

$$z_i = 2 * (x_i + x_i * y_i + y_i)$$

main aðferðin les inn af staðalinntaki hámarksfjölda (**fjoldiStaka**) staka í fylkjum **x**, **y** og **z**. Ef staðalinntakið inniheldur færri en **fjoldiStaka** para þá eru restin af stökunum 0.

Forritið hluta af **main** aðferðinni sem býr til fylkin **x**, **y** og **z** les af staðalinntaki heiltölur og setur þau í fylkin **x** og **y**,

restin af **main** aðferðinni reiknar **z** og prentar **z**.

Sem dæmi ef notandi skrifar á console eftirfarandi gögn og **endar með ctrl D** til að enda innsláttinn:

```
3      <-- inntak
1 1
3 5
^D (ctrl D)
[ 6, 46, 0] <-- úttak
```

Þá væru fylkin **x** [1, 3, 0] og **y** [1, 5, 0] og **z** [6, 46, 0]

Sjá frekari prófunartilvik í Test code

Ef þú getur ekki keyrt forritið eða færð villu, gerðu þitt besta og við förum handvirkt yfir forritið samkvæmt matskvarða

7

Forritið og prófið forritið

Test case #	Input	Expected output
1	3 1 1 3 5	[6, 46, 0]
2	4 3 2 1 5 4 1 2 0	[22, 22, 18, 4]
3	1 3 7	[62]

```
import java.util.Arrays;
import java.util.Scanner;
class Formula {
    // forrita reiknaFormula hédan
    public static ... {

    }
    // reiknaFormula endar hér
    public static void main(String[] args) {
        Scanner inntak = new Scanner(System.in, "UTF-8");
        int fjoldiStaka = inntak.nextInt();
        // forrita hédan

        // forrita hingað
        reiknaFormula(x, y, z);
        System.out.println(Arrays.toString(z));
    }
}
```

```
public static void reiknaFormula(int[] x, int[] y, int z[]) {
    for (int i = 0; i < x.length; i++) {
        z[i] = 2 * (x[i] + x[i] * y[i] + y[i]);
    }
}
```

```
public static void main(String[] args) {
    Scanner inntak = new Scanner(System.in, "UTF-8");
    int fjoldiStaka = inntak.nextInt();
    // forrita hédan
    int[] x = new int[fjoldiStaka];
    int[] y = new int[fjoldiStaka];
    int[] z = new int[fjoldiStaka];
    for (int i = 0; i < fjoldiStaka && inntak.hasNext(); i++) {
        x[i] = inntak.nextInt();
        y[i] = inntak.nextInt();
    }
    // forrita hingað
    reiknaFormula(x, y, z);
    System.out.println(Arrays.toString(z));
}
```

Test code

Maximum marks: 16

Gefið ykkur að til sé klasinn **WriteableObject** sem hefur eftirfarandi aðferð skilgreinda en tóma

```
public void write(PrintWriter writer)
```

Til er klasinn **Circle** sem erfir (extends) frá **WriteableObject** klasanum. **Circle** hefur tilvikið **radius** sem er af taginu **int**. Til er klasinn **ClientWriteObjects** sem getur skrifað í skrá **Circle** hluti og aðra hluti sem eru af klasa sem erfa (extends) frá **WriteableObject** (t.d. Rectangle)

Ljúkið við að forrita aðferðirnar hér á eftir

a) write í Circle

b) writeObjects í ClientWriteObjects og

c) main í ClientWriteObjects.

Afritið write, writeObjects og main aðferðirnar í java ritilinn (e. editor)

Þið þurfið **ekki** að skrifa import setningar. Þið þurfið **ekki** að grípa eða skilgreina **exceptions**.

```
import java.io.PrintWriter;
```

```
// Circle
```

```
public class Circle extends WriteableObject {
```

```
    private int radius;
```

```
    /**
```

```
     * Smíðar hring með radíus
```

```
     * @param radius radíus
```

```
     */
```

```
    public Circle(int radius) {
```

```
        this.radius = radius;
```

```
    }
```

```
    public void write(PrintWriter minnWriter) {
```

```
        // Forritið héðan og
```

```
        // skrifið hlutinn (this) á minnWriter á sniðinu "Circle <radius>" í sér línu
```

```
        // t.d. ef radius hlutarins er 3 þá er prentað "Circle 3"
```

```
        // hingað
```

```
    }
```

```
}
```

```
import java.util.ArrayList;
```

```
import java.io.PrintWriter;
```

```
public class ClientWriteObjects {
```

```
    public static void writeObjects(ArrayList<WriteableObject> listOfObjects,
                                    PrintWriter minnWriter) {
```

```
        // Forritið héðan og
```

```
        // skrifið út alla hlutina í listOfObjects á minnWriter (notaðu write aðferðina)
```

```
        // hingað
```

```
    }
```

```
    public static void main(String[] args) {
```

```
        // listi af Circle og Rectangle hlutum
```

```
        ArrayList<WriteableObject> list = populateList();
```

// Forritið héðan og

```
// Búið til skráar object með skráarnafni sem er í args[0].  
// Búið til prentara object svo hægt sé að skrifa í skrána  
// kallið á writeObjects aðferðina með viðeigandi parametrum  
// lokið prentara object
```

// hingað

```
}
```

```
public void write(PrintWriter writer) {  
    // skrifið hlutinn á writer á sniðinu "Circle <radius>"  
    // ef radius hlutarins er 3 þá er prentað "Circle 3"  
    writer.println("Circle " + radius);  
}
```

```
public static void writeObjects(ArrayList<WriteableObject> listOfObjects,  
                                PrintWriter writer) {  
    // skrifið út alla hlutina í listOfObjects á writer  
    for (WriteableObject g : listOfObjects) {  
        g.write(writer);  
    }  
}
```

```
public static void main(String[] args) throws IOException {  
    // listi af Circle og Rectangle hlutum  
    ArrayList<WriteableObject> list = populateList();  
    // Búið til skráar object með skráarnafni sem er í args[0].  
    File file = new File (args[0]);  
    // Búið til prentara object svo hægt sé að skrifa í skrána  
    PrintWriter minnWriter = new PrintWriter (file, StandardCharsets.UTF_8);  
    // kallið á writeObjects aðferðina með viðeigandi parametrum  
    writeObjects(list, minnWriter);  
    // lokið prentara object  
    minnWriter.close();  
}
```

8

Forritið

1	
---	--

Maximum marks: 16

Klasinn **Ferd** inniheldur tvo eiginleika ferðar, verð flugs og verð gistingar sem eru geymd í tilviksbreytum. Klasinn hefur smíð sem tekur inn tvo parametra, verð flugs og verð gistingar. Beinagrind af klasanum er gefinn hér á eftir

```
public class Ferd {
    private double flug;
    private double gisting;
    public Ferd(double flug, double gisting) {
        this.flug = flug;
        this.gisting = gisting;
    }
    public double getFlug() {
        return flug;
    }
    public double getGisting() {
        return gisting;
    }
}

public static double verdmunur(Ferd ferd1, Ferd ferd2) {
    // Forritið héðan

    // og hingað
}

public Ferd medalFerd(Ferd ferd) {
    // Forritið héðan

    // og hingað
}

public static void main(String[] args) {
}
}
```

A. Forritið aðferðina **verdmunur** sem er reiknaður á eftirfarandi hátt ef **flug1** er verð flugs í **ferd1** og **flug2** er verð flugs í **ferd2** og sama með gistingu.

$$verdmunur = \sqrt{(flug2 - flug1)^2 + (gisting2 - gisting1)^2}$$

B. Forritið aðferðina **medalFerd** sem skilar **nýrri Ferd** með meðaltali tveggja fluga og meðaltali tveggja gistinga.

9

Forritið hér

1

```
public static double verdmunur(Ferd ferd1, Ferd ferd2) {  
    return Math.sqrt(Math.pow(ferd2.getFlug() - ferd1.getFlug(), 2)  
        + Math.pow(ferd2.getGisting() - ferd1.getGisting(), 2));  
}  
  
public Ferd medalFerd(Ferd ferd) {  
    return new Ferd((this.getFlug() + ferd.getFlug()) / 2,  
        (this.getGisting() + ferd.getGisting()) / 2);  
}
```

Maximum marks: 30