VSTUPNÍ JÍDELNÍČEK

Měsíc a rok

Ing. Eva Hanzalová

+420 725 880 350 eva.hanzalova@provyziva.cz

VÝPOČTY

Pokud není uvedeno jinak, jsou všechny hodnoty v jídelníčku uvedeny **v syrovém stavu**. V případě vážení hotového pokrmu, je třeba hodnoty přepočítat následovně.

Přílohy

Okopaniny a výrobky z nich

Brambory	
vařené	beze změny
pečené	hmotnost x 0,5

Zrniny a výrobky z nich

Rýže, těstoviny apod.	
vařené	x 2,5

Bílkoviny

Maso	
vařené	x 0,9
pečené	x 0,8
dušené	x 0,7

Skladba jídelníčku

Vždy skládáme jídlo ze 3 skupin – sacharidy, bílkoviny a tuky (u snídaně i ovoce). Pokud nezapomene na žádnou potravinu z těchto skupin, pak zvládneme složit například jogurt s granolou, oříšky a sušeným ovocem nebo klidně svíčkovou k obědu (bez zahušťovadel) 😊

Kombinace potravin v rámci jedné skupiny

.. pokud budeš chtít k jídlu např. šunku a současně i sýr, rozděl dané suroviny ve vzájemném poměru tak, aby množství sedělo (např. v poměru **1:1** by to mohlo znamenat **_ g šunky** a **_ g sýra**, pokud by byla řeč o snídani).

PITNÝ REŽIM

Voda a neslazené nápoje

Voda	
Čaj	
Prolinie a další nápoje s umělými sladidly**	
	alespoň **2 litry denně**

*ráno po probuzení vypít 1 skleničku vody / čaje

**umělá sladidla nejsou nejvhodnější alternativou, doporučuji zařazovat pouze příležitostně

Káva

Káva s mlékem	max. **2 – 3 šálky denně**

*mléko přidané v kávě se do denního počtu nepočítá

*cukr používat co nejméně, nicméně 1 lžička cukru je vhodnější varianta než celý zákusek 😊

KOMENTÁŘ: Takhle by to mělo vypadat, ale kávu do denního součtu raději nepočítáme :)

SNÍDANĚ

Sacharidová složka

Pečivo

Chléb	
pšeničnožitný a žitnopšeničný	_ - _ g
toustový celozrnný	
celozrnný	
vícezrnný	
Běžné pečivo	
cereální (kaiserka)	
žitné	
celozrnné	
vícezrnné	
grahamové	
Speciální a trvanlivé pečivo	
bramborové	
kukuřičné	
pita chléb (tmavý)	
tortilla celozrnná	
sušené (knäckebrot, kukuřičné chlebíčky aj.)	
"pufované" pečivo (rýžové a jiné chlebíčky)	
bezkvasové (macesy)	

Vločky, müsli, granola

müsli neochucené (vločky)	
müsli ochucené (s ovocem, čokoládou, oříšky)	
granola*	

Zrniny

rýže, pohanka, špalda	
quinoa	
amarant	

*granola má poměrně vysoký obsah cukru a tuku, proto je množství adekvátně sníženo

Formula

1. Calculate relative atomic mass of Cu

$$A_r(Cu) = 63.546$$

2. Calculate molar mass of Cu

$$M(Cu) = A_r(Cu) \cdot 1\ g/mol = 63.546\ g/mol$$

3. Calculate number of moles

$$n(Cu) = \frac{m(Cu)}{M(Cu)} = \frac{0.635\ g}{63.546\ g/mol} = 0.01\ mol$$

4. Calculate mass of CuO

$$m(CuO) = n(CuO) \cdot M(CuO) = 0.01\ mol \cdot 79.546\ g/mol = 0.7955\ g$$

5. Calculate number of moles

$$n(CuO) = 0.01\ mol$$

Calculate

$$n(CuO) = \frac{m(CuO)}{M(CuO)}$$

$$M(CuO) = A_r(Cu) + A_r(O) = 63.546 + 16 = 79.546\ g/mol$$

$$m(CuO) = n(CuO) \cdot M(CuO) = 0.01\ mol \cdot 79.546\ g/mol = 0.7955\ g$$

Calculate mass of CuO

$$m(CuO) = 0.7955\ g$$

Calculate mass of Cu

$$m(Cu) = m(CuO) \cdot \frac{A_r(Cu)}{M(CuO)} = 0.7955\ g \cdot \frac{63.546\ g/mol}{79.546\ g/mol} = 0.635\ g$$

Final answer: $m(Cu) = 0.635\ g$

Formula: $m(Cu) = m(CuO) \cdot \frac{A_r(Cu)}{M(CuO)}$, $M(CuO) = A_r(Cu) + A_r(O)$, $n(CuO) = \frac{m(CuO)}{M(CuO)}$, $m(CuO) = n(CuO) \cdot M(CuO)$, $n(Cu) = \frac{m(Cu)}{M(Cu)}$, $A_r(Cu) = 63.546$, $M(Cu) = A_r(Cu) \cdot 1\ g/mol$

Bílkovinná složka

Maso a masné výrobky

šunka dušená / vařená
šunka sušená
párky (80 % masa a více)

Mléko a mléčné výrobky, vejce

mléko
jogurt bílý
tvaroh

Sýry zrající
všechny typy plátkových sýrů*

Sýry nezrající
cottage
mozzarella

Vejce
slepičí (velikost m)
křepelčí
pouze bílek

*U mléčných výrobků jako jsou jogurty či tvaroh zvol spíše tučnější varianty – pomůže nám to adekvátně snížit příjem přidaných tuků a tím i objem, viz. dále

POZOR č.1: Bílkoviny obsahují skryté tuky, se kterými je nutno dále počítat. Proto se budou hodnoty přidaných tuků a olejů měnit podle toho, zda zvolená bílkovina obsahuje **malé množství**, **střední množství**, či **vysoké množství** tuku dle následující tabulky.

POZOR č. 2: **Ne všechny potraviny jsou zde v jídelníčku označené** – je potřeba aby k nim bylo přistupováno podle toho, jak vysoký obsah tuku mají (např. **libová kuřecí šunka** vs. **prorostlá šunka** nebo **sýr s 15 % tuku** vs. **sýr s 45 % tuku**).

Množství přidaného tuku ze sekce „Tuky a oleje“

Potravina obsahuje nízké množství tuku	**hmotnost x 1**
Potravina obsahuje střední množství tuku, či celé jídlo obsahu kombinaci tuku nízkého a vysokého	**x 0,5**
Potravina obsahuje vysoké množství tuku	**x 0,3**

Tuky a oleje

Živočišné tuky

máslo klasické
máslo pomazánkové
lučina 70 %
gervais original, žervé lučina a jiné mazací sýry**

Rostlinné tuky a oleje

avokádo

Oleje
na teplou kuchyni doporučuji řepkový olej
na studenou kuchyni doporučuji extra panenský olivový olej

Ořechy a ořechová másla, semínka
vlašské, piniové, lískové ořechy
arašídy, mandle, pistácie
lněné semínko
chia semínka

**jedná se sice o sýry, z důvodu vysokého obsahu tuku je však budeme řadit do této skupiny potravin

*** **nezapomeň přizpůsobit množství dle návodu na předchozí stránce** 😊 Např. u tučného tvarohu pak přidáš pouze 15 g vlašských ořechů (30 % přidaného tuku).

OBĚD

Sacharidová složka

Pečivo

Chléb

pšeničnožitný a žitnopšeničný
toustový celozrnný
celozrnný
vícezrnný

Běžné pečivo

cereální (kaiserka)
žitné
celozrnné
vícezrnné
grahamové

Speciální a trvanlivé pečivo

bramborové
kukuřičné
pita chléb (tmavý)
tortilla celozrnná
sušené (knäckebrot, kukuřičné chlebíčky aj.)
"pufované" pečivo (rýžové a jiné chlebíčky)
bezkvasové (macesy)

Okopaniny

Brambory a výrobky z nich

brambory
bramborová kaše (po uvaření)
bramborové knedlíky, noky, halušky (po uvaření)

Batáty a výrobky z nich

batáty
batátová kaše

Zrniny

rýže, pohanka, špalda
quinoa
amarant

[illegible]

[illegible]

[illegible]

[

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]
[illegible]
[illegible]
[illegible]

[illegible]

[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]

[illegible]

[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]

[illegible]

[illegible]

[illegible]
[illegible]
[illegible]
[illegible]

[illegible]

[illegible]
[illegible]
[illegible]

[illegible]

[illegible]

[illegible]
[illegible]
[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

Effective Java

Item 1: Consider static factory methods instead of constructors

```
public static Boolean valueOf(boolean b) {
    return b ? Boolean.TRUE : Boolean.FALSE;
}
```

```
private static Service[] services = new Service[4];
private static int numProviders = 0;

public static void registerProvider(Service s) {
    services[numProviders++] = s;
}
```

Item 2: Consider a builder when faced with many constructor parameters

```
NutritionFacts cocaCola = new NutritionFacts(240, 8, 100, 0, 35, 27);
```

```
NutritionFacts cocaCola = new NutritionFacts.Builder(240, 8)
    .calories(100)
    .sodium(35)
    .carbohydrates(27);
```

```
public class NutritionFacts {
    int servingSize;
    int servings;
    int calories;
    int fat;
    int sodium;
    int carbohydrates;

    public NutritionFacts(int servingSize, int servings,
        int calories, int fat,
        int sodium, int carbohydrates) {
        this.servingSize = servingSize;
        this.servings = servings;
        this.calories = calories;
        this.fat = fat;
        this.sodium = sodium;
        this.carbohydrates = carbohydrates;
    }
}
```

```
public class NutritionFacts {
    int servingSize;
    int servings;
    int calories;
    int fat;
    int sodium;
    int carbohydrates;

    public static class Builder {
        int servingSize;
        int servings;
        int calories = 0;
        int fat = 0;
        int sodium = 0;
        int carbohydrates = 0;

        public Builder(int servingSize, int servings) {
            this.servingSize = servingSize;
            this.servings = servings;
        }

        public Builder calories(int val) {
            calories = val;
            return this;
        }

        public Builder sodium(int val) {
            sodium = val;
            return this;
        }

        public Builder carbohydrates(int val) {
            carbohydrates = val;
            return this;
        }

        public NutritionFacts build() {
            return new NutritionFacts(this);
        }
    }

    private NutritionFacts(Builder b) {
        servingSize = b.servingSize;
        servings = b.servings;
        calories = b.calories;
        fat = b.fat;
        sodium = b.sodium;
        carbohydrates = b.carbohydrates;
    }
}
```

Item 3: Enforce the singleton property with a private constructor

```
public class Elvis {
    public static Elvis INSTANCE = new Elvis();
    private Elvis() {}

    public void leaveTheBuilding() {}
}
```

```
public class Elvis {
    private static Elvis[] INSTANCES = {new Elvis(), new Elvis()};
    private int i = 0;
    private Elvis() {}

    public static Elvis getInstance() {
        return INSTANCES[i++];
    }

    public void leaveTheBuilding() {}
}
```

```
enum Elvis {
    INSTANCE;

    public void leaveTheBuilding() {}
}
```

Item 4: Use a private constructor to enforce noninstantiability

```
public class UtilityClass {
    private UtilityClass() {}

    public static void foo() {}
}
```

```
public class UtilityClass {
    private UtilityClass() {
        throw new AssertionError();
    }

    public static void foo() {}
}
```

Item 5: Avoid creating unnecessary objects

```
String s = new String("stringette");
```

```
boolean[] primes = new boolean[1000];
for (int i = 2; i < 1000; i++)
    primes[i] = true;
// Remove multiples of primes starting from 2
for (int i = 2; i * i < 1000; i++)
    for (int j = i * i; j < 1000; j += i)
        primes[j] = false;
```

```
public class Person {
    private Date birthDate;

    public boolean isBabyBoomer() {
        Calendar gmtCal = Calendar.getInstance(TimeZone.getTimeZone("GMT"));
        int year = gmtCal.get(Calendar.YEAR);
        int month = gmtCal.get(Calendar.MONTH);
        int day = gmtCal.get(Calendar.DAY_OF_MONTH);
        Date start = gmtCal.getTime();
        gmtCal.set(year, month, day, 18, 0, 0);
        Date end = gmtCal.getTime();
        return birthDate.compareTo(start) >= 0
            && birthDate.compareTo(end) < 0;
    }
}
```

Item 6: Eliminate obsolete object references

```
public class Stack {
    private Object[] elements;
    private int size = 0;

    public Object pop() {
        if (size == 0)
            throw new EmptyStackException();
        Object result = elements[--size];
        elements[size] = null;
        return result;
    }
}
```

```
import java.util.*;

public class Cache {
    private Map<Object, Object> cache = new HashMap<Object, Object>();

    public void put(Object key, Object value) {
        cache.put(key, value);
    }

    public Object get(Object key) {
        return cache.get(key);
    }
}
```

Item 7: Avoid finalizers

```
public class Foo {
    protected void finalize() {
        System.out.println("Finalizing");
        super.finalize();
    }
}
```

Item 8: Prefer enum types to int constants

```
public class Apple {
    public static int FUJI = 0;
    public static int PINK_LADY = 1;
    public static int GRANNY_SMITH = 2;
}
```

```
enum Apple {
    FUJI,
    PINK_LADY,
    GRANNY_SMITH;
}
```

```
enum Apple {
    FUJI("Fuji"),
    PINK_LADY("Pink Lady"),
    GRANNY_SMITH("Granny Smith");

    private String name;

    Apple(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }
}
```

The enum type offers compact, robust implementations of singleton classes, static utility classes, and classes that implement one or more interfaces. Many of the advantages of enum types are invisible to client programs, but they make it easy to write powerful, flexible, and robust programs. Enum types are not appropriate for all situations, but they are more than a match for most of the situations in which you need to write your own type. Enum types are a good choice for any set of related constants whose members are known at compile time. Examples include the planets in the solar system, the days of the week, the coins in the U.S. currency system, and the card suits in a game.

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]