Studienleistung Programmieren 2 Teil 1

- **Was sind/wozu dienen "Namespaces" in Programmiersprachen?**

  ein Namespace ist eine logische Namenskonvention für das Abgrenzen von Klassen und sonstigen Typen, um die Gefahr von Kollisionen infolge mehrfach verwendeter und somit nicht mehr eindeutigen Bezeichner-Namen zu vermeiden. Die .NET-Klasse *MessageBox* wird zum Beispiel im Namespace *System.Windows.Forms* deklariert und implementiert.

- **Wie lautet die Syntax zu Namespaces in Java und C# (Beispiel)**

  Java:

  ```
  package harding.compsci.graphics;
  ```

  C#:

  ```
  namespace Harding.Compsci.Graphics {
      …
  }
  ```

- **Wie unterscheiden sich Namespaces in C# und Java?**

  Java Packages können nicht verschachtelt werden. Eine Datei kann nur ein Package enthalten.

- **Wie werden Arrays in Java/C# erzeugt (Beispiel)? Sind diese Werte-, oder Referenztypen - was würde also bei einer Zuweisung eines Arrays, zu einem anderen passieren?**

  Java:

  ```
  int nums[] = {1, 2, 3};   or   int[] nums = {1, 2, 3};
  ```

  C#:

  ```
  int[] nums = {1, 2, 3};
  ```

  Es sind Referenztypen. Bei der einer zuweisug würde den beide Variablen auf das gleiche Array zeigen.

- **Geben Sie Beispiele für Wertetypen und Referenztypen in C# und Java.**

  Wertetypen integer, boolean

  Referenztypen Arrays, Objekte

- **Definieren Sie: Datenabstraktion und Information Hiding (Ziele des Einsatzes von Klassen), Eigenschaft/Attribut, Methode, statische Methode, statische Eigenschaft, statische Klasse, Wertetyp,Referenztyp**

  Information Hiding: einige Informationen werden anderen Teilen des Programms vorenthalten, da es nicht relevant ist die Informationen zu teilen.

  Attribut: Eigenschaft eines Objekts.

  Methode: eine Funktion eines Objekts.

  statische Methode: methode die aufgerufen werden kann ohne das die Klasse instantiiert werden muss.

  statisches Attribut: das gleiche wie methoden nur für Attribute

  statische Klasse: Klasse die nicht instantiiert werden kann.

  Wertetyp: Datentyp der bei einer Zuweisung kopiert werden muss.

  Referenztyp: Datentyp beidem bei einer Zuweisung nur eine Referenz auf den gleichen Speicher übergeben wird. Alle variablen zeigen immer auf die gleichen Daten.

- **Was ist der Unterschied zwischen "ausgefransten" (unregelmäßigen) Arrays und rechteckigen (regelmäßigen Arrays)? Welche Arten werden in Java/C# unterstützt? Beispiel! Gibt es Vor-/Nachteile?**

  Mehrdimensionale Arrays verfügen über mehr als eine Dimension.

  Ein unregelmäßiges Array ist lediglich ein Array von Arrays und wird als »unregelmäßig« (jagged) bezeichnet, da es nicht zwangsläufig rechteckig ist.

  beide arten werden in Java und C# unterstützt.

- **Schreiben Sie ein Programm, welches 2 Namenräume A und B enthält. Deklarieren Sie eine Klasse "Zeit" (vorerst keine speziellen "Fähigkeiten" der Klasse notwendig) im Namenraum A und verwendet Sie diese (erzeugen Sie ein Objekt) im Namensraum B.**

- **Erweitern Sie die Klasse "Zeit" zur Verwaltung von Uhrzeiten. Intern sollen Uhrzeiten in Sekunden, Minuten und Stunden gespeichert werden. Implementieren Sie Getter/Setter für diese drei Eigenschaften. Sehen Sie auch überladene Konstruktoren vor um die Eigenschaften eines Objekts direkt bei seiner Initiierung zu setzen.**

  sp_prog2_e1.tar

- **In C# gibt es einen "struct" Typ. Diskutieren Sie: Wann ist der Einsatz einer Klasse, wann der einer Struktur sinnvoll?** Der Einsatz einer Struktur ist immer dann sinnvoll wenn keine Methoden benötigt werden.
- **In Java können Operatoren nicht überladen werden (zumindest nicht durch den Entwickler, intern geschieht das durchaus, nämlich bei der Zeichenfolgenverkettungen). Welche Gründe könnte es geben, dass dies in Java nie realisiert wurde?**

  Weil sie nicht wollten. Und weil das gleiche Ziel auch mit einer neuen Methode erreicht werden kann.
- **Sind Stringzuweisungen Zeigerzuweisungen, oder Wertezuweisungen? D.h. wird der Wert des Strings kopiert, oder nicht? Wenn dies z.B. in einer Sprache keine Kopie erzeugt, wie wäre dann eine Kopie zu erzeugen?**

  C#:

  Das String object wir zu einem datentyp simuliert und somit handelt es sich um eine Wertezuweisung.

  Java:

  Es handelt sich um ein object und somit um eine Referenzzuweisung. Um einen String zu kopieren muss ein neues String object erzeugt werden und der existirende String als parameter Übergeben werden.
- **Sind die Opertoren == und != Wertevergleiche? Wenn dies für eine Sprache nicht gilt, wie ist dann das Vorgehen bei Vergleichen?**

  Ja die operatoren sind werte vergleichend. Da es sich bei Java Strings jedoch um Objekte handelt können die Strings nicht verglichen werden, da der Operator lediglich die speicher referenzen auf die jeweiligen String Objekte vergleicht.
- **Sind die Operatoren <, <= und >, >= erlaubt? Wenn nicht, wie können entsprechende Vergleiche getätigt werden?**

  Nein da diese Opertoren auf mathematische Größe vergleichen. Möchte man die länge eines Strings überprüfen so kann das mit der length() methode umgesetzt werde.
- **Lesen Sie die Beschreibung der Klasse StringBuilder. Wann/Warum ist der Einsatz dieser Klasse anstelle der Stringverkettung zu empfehlen?**

  Der einsatz ist immer dann sinvoll wenn sehr viele strings mit einanner verbunden werden sollen, das der StringBuilder um einiges schneller ist.
- **Sind diese Grundsätze zur Parameterübergabe auch in C# gültig?**

  Nein, parameter können in C# auch als referenzen übergeben werden. Sie werden standartmäßig jedoch als wert übergeben.

- **In C# gibt es die Schlüsselwörter "ref" und "out". Erklären Sie anhand eines Experiments die Funktion der beiden Schlüsselwörter.**

  Um einer Funktion eine variable als referenz zu übergeben, kann das schlüsselwort ref verwendet werden. Wird out verwendet, so muss die Variablen bei der übergabe nicht initialisiert sein.

- **In Java gibt es die beiden obigen Schlüsselwörter nicht. Simulieren Sie das Verhalten von "ref", in dem Sie den zu übergebenden String das Objekt einer Wrapper-Klasse einbetten und das Wrapper Objekt an die Funktion "giveMeAString" übergeben. --> Erläutern Sie das Ergebnis.**

```
class Wrapper{
    private Object value;

    public Wraper(value){
        this.value= value;
    }

    public Object get(){
        return value;
    }

    public void set(value){
        this.value= value;
    }
}

class Test{
    public static void main(String[] args){
        Object x = new Wrapper(null);
        giveMeAString (x);
        System.out.println (x.get());
        [...]
    }

    static void giveMeAString (Object y){
        y.set("This is a string");
    }
}
```

  Das string object wird nun in einer Wrapper klasse gespeichert. Dieses Wrapper objekt wird dann in an die funktion übergeben und diese ändert denn inhalt. Da es sich anschließend jedoch immer noch um das gleiche Wrapper Objekt handelt kann der neue string immernoch zugegriffen werden.