**Strukturierte Vorgehensweise bei der Erstellung von Programmen**

Bei der Form-Programmierung sollte man folgende Schritte anwenden:

1. **Arbeitsschritte überlegen**

**2. Form erstellen**

**3. Objekte umbenennen**

**4. Variablen deklarieren und initialisieren**

**5. Ereignis erzeugen**

**6. Arbeitsschritte in Quellcode übersetzen**

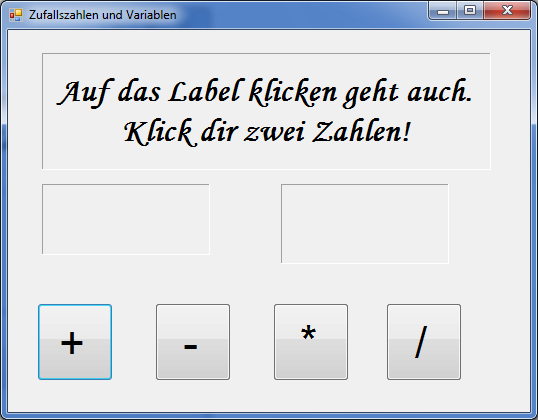
**7. Teilfunktionen testen**

**8. Gesamtfunktion testen und verbessern**

Die Schritte 6, 7 und 8 müssen dabei immer mehrfach durchlaufen werden.

**Arbeitsauftrag:**

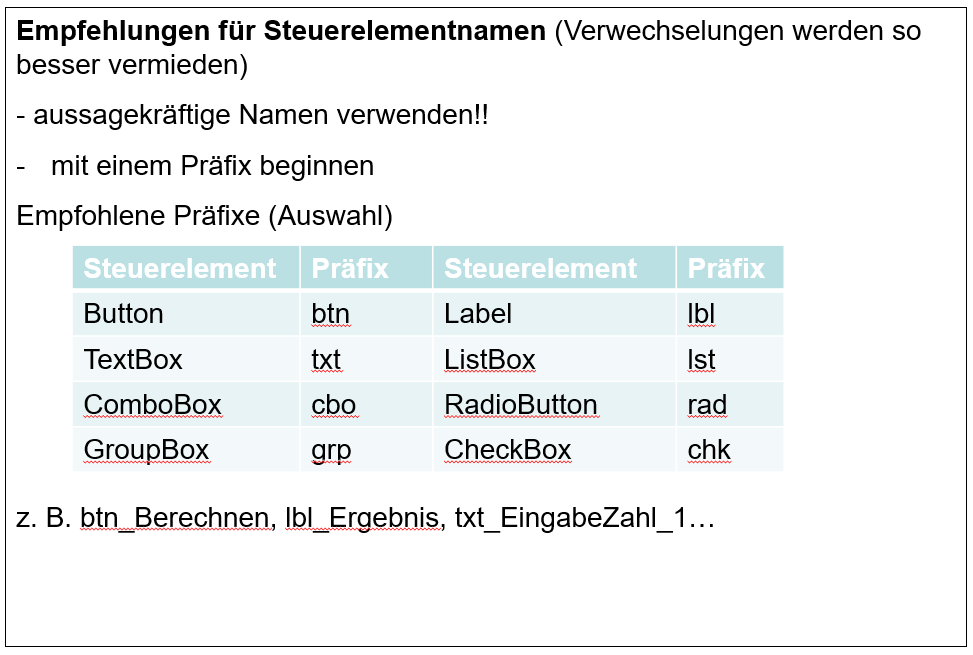
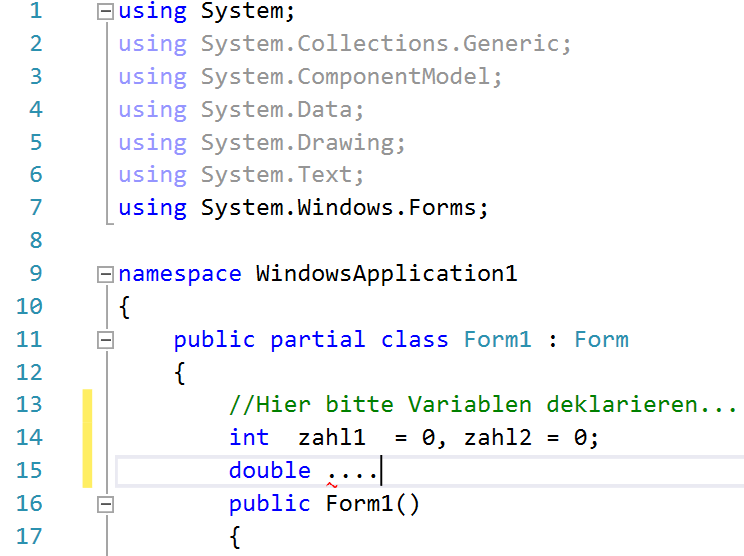
1. **Form erstellen**: Erstellt die dargestellte Form. Berücksichtigt dabei die einzustellenden Eigenschaften der Objekte aus der Tabelle.



**Zahl2**

**Zahl1**

|  |  |  |
| --- | --- | --- |
| **Objekt** | **Eigenschaft** | **Wert** |
| Form1 | Text | Zufallszahlen und  Variablen |
| button1…4 | Text | +, -, \*,/ |
| Label1 für Anfangstext und Ergebenis | Text  Borderstyle  Autosize | Auf das …  Fixed3D false |
| Label 2-3 für Zahl 1  und Zahl 2 | Borderstyle  Autosize | Fixed3D  false |

1. **Objekte umbenennen:** Nachdem die Form designt ist, müssen die Objekte umbenannt werden. Dabei sollen sinnvolle Namen benutzt werden, welche dem Programmierer klarmachen um was für eine Objekt es sich handelt und welche Bedeutung es hat.  
   Einen Button1 umbenennen in „plus“ wäre nur die halbe Lösung, da man im Programmiercode nicht erkennen würde, dass es sich um einen Button handelt.   
   Ein sinnvoller Name wäre „btn\_plus“.  
   Vergebt Namen für die sieben Objekte berücksichtigt dabei die Empfehlungen.  
     
   Check
2. **Variablen deklarieren und initialisieren:** Man überlegt sich wo benötige ich Variablen (benannte Speicher) und was für ein Typ müssen sie sein.  
   Bei unserem Beispiel bedeutet das sicherlich das ich drei Variablen brauche (z. B. zahl1, zahl2 und Ergebnis) Die Anzahl der Variablen kann jederzeit geändert werden.  
   Da erstmal nur ganze Zahlen erzeugt werden sollen ist es sinnvoll, das „zahl1“ und „zahl2“ vom Integer-Typ (int) und das Ergebnis vom Double-Typ (double).  
   Variablen können an verschiedenen Stellen im Program deklariert werden. Nehmt bitte zu Beginn die markierte Position.  
   
3. **Ereignis erzeugen:** Das Ereignis „doppelclick“ ist am einfachsten zu erzeugen in dem man auf das jeweilige Objekt (Button oder Label) einen Doppelclick anwendet. Es wird ein neuer Reiter mit dem entstanden Code aufgemacht und folgender Code angzeigt.   
   private void lbl\_ZahlenErzeugen\_Click(object sender, EventArgs e)  
   { //Der Programmiercode der hier steht wird bei btn\_Plus\_Click ausgeführt.  
   }
4. **Arbeitsschritte überlegen:** Ihr müsst kleinschrittig überlegen was das Programm (der PC) bei diesem Ereignis machen soll.   
   z. B. lbl\_ZahlenErzeugen beim Doppelclick auf das Label müssen folgende Schritte durchgeführt werden:  
    \* Zufallsgenerator starten  
    \* Zufallszahl 1 erzeugen und einer Variablen (Zufallszahl1) zuweisen  
    \* Zufallszahl2 erzeugen und einer Variablen (Zufallszahl2) zuweisen  
    \* Zufallszahl1 in einem Label ausgeben  
    \* Zufallszahl2 in einem Label ausgeben
5. **Arbeitsschritte in Quellcode übersetzen:** Jetzt muss man mit Hilfe der angegebenen Befehle den Quellcode schreiben:  
   Random Zufall = new Random();   
    // (Zufallsgenerator starten (einfach ausgedrückt))  
   Zufallszahl1 = Zufall.Next(1,10);    
    //(Zufallszahl 1 erzeugen und einer Variablen zuweisen)  
   label1.Text= “Zufallszahl 1 = “ + Zufallszahl1.ToString()  
    //Ausgabe des Textes (Zufallszahl 1 + Wert; Konvertierung))
6. **Teilfunktionen testen:** Nach einem kleinen Programmierabschnitt die Teilfunktionen bzw. Befehle testen und mit Hilfe der Fehleranzeige den Fehler beheben.
7. **Gesamtfunktion testen und verbessern**

#### Kommentar

**Syntax:**

//einzeiliger Kommentar. Wird vom Debugger ignoriert!

und

/\*

mehrzeiliger Kommentar er wird auch vom Debugger ignoriert!

Alles was zwischen den Sternen steht wird vom Compiler ignoriert.

\*/

**Erklärung:**

Kommentare werden benötigt um Erklärungen im Programm zu geben.

Außerdem sind mehrzeilige Kommentare wichtig bei der Suche nach Fehlern. Mit den beiden Zeichen können Programmcodesabschnitte deaktiviert werden.

**Zuweisung**

Mit Zuweisungen erhalten [Variablen](https://moodle.bbs-me.de/Moodle/mod/glossary/showentry.php?courseid=110&eid=248&displayformat=dictionary) einen Wert. Auf der linken Seite vom Gleichheitszeichen muss die [Variable](https://moodle.bbs-me.de/Moodle/mod/glossary/showentry.php?courseid=110&eid=248&displayformat=dictionary) stehen. Auf der rechten Seite können verschiedene Ausdrücke stehen.

**Befehl:**

[Variable](https://moodle.bbs-me.de/Moodle/mod/glossary/showentry.php?courseid=110&eid=248&displayformat=dictionary) = Wert der zugewiesen werden soll, steht auf der rechten Seite vom Gleichheitszeichen.

**Programmierbeispiele:**

Zahl1 = 0; // Zahl1 wird der Wert 0 zugewiesen.

Summe= Zahl1 + Zahl2; // Der Variablen "Summe" wird die Summe von Zahl1 und Zahl2 zugewiesen

Zahl1 = [Convert](https://moodle.bbs-me.de/Moodle/mod/glossary/showentry.php?courseid=110&eid=258&displayformat=dictionary).ToDouble(textBox1.Text); // Der Text aus textBox1 wird in eine double-Zahl konvertiert und der Zahl1 zugwiesen.

Zahl1 = [Zufall](https://moodle.bbs-me.de/Moodle/mod/glossary/showentry.php?courseid=110&eid=259&displayformat=dictionary).Next(100); // Next ist eine [Methode](https://moodle.bbs-me.de/Moodle/mod/glossary/showentry.php?courseid=110&eid=366&displayformat=dictionary) von [Random](https://moodle.bbs-me.de/Moodle/mod/glossary/showentry.php?courseid=110&eid=259&displayformat=dictionary) (Es wird eine Ganzzahl zwischen 0 und kleiner 100 generiert und der [Variablen](https://moodle.bbs-me.de/Moodle/mod/glossary/showentry.php?courseid=110&eid=248&displayformat=dictionary) zugewiesen)

|  |
| --- |
| Random **Syntax:**  Random Zufall = new Random();  int zahl = Zufall.Next(min,max);    (zufällige Zahl zwischen min und max)  **Erklärung:**  Der Befehl "Random" generiert eine zufällige Zahl zwischen einem minimalen und einem maximalen Wert. Dabei muss man beachten, dass die maximale Zahl nicht verwendet wird.  "rnd" und "Zahl" gehören nicht zur Syntax und können dementsprechend durch einen beliebigen Namen ersetzt werden.  Zahl1 = Zufall.Next ();                        .Next ()            Liefert int-Z\_zahl        0 <= zahl < 2.147483.647  .Next (n)           Liefert eine int-Z\_zahl    0 <= zahl < n                    .Next (n1, n2)      Liefert eine int-Z\_zahl   n1 <= zahl < n2                    .NextDouble() Liefert eine double-Z\_zahl        0  <= zahl < 1  **Beispiel:**  Generieren eines zufälligen Wertes zwischen 1 und 100:  Random Zufall = new Random();  zahl = Zufall.Next(1,101);   //zahl wird der Wert zugewiesen  label1.Text = zahl.ToString();          //zahl wird auf dem Label ausgegeben  **Ausgabe**  **Syntax:**  **.To*Dateityp()***  **Erklärung:**  Was **in vor dem Punkt steht, wird in einen anderen Dateityp, der vor den Klammern steht, umgewandelt. Z. B. Zahl1.ToString()**  **Die Variable Zahl1 wird in einen Text gewandelt.**  **Beispiel:**  Mittelwert = (a + b) / 2;  label1.Text="Mittelwert ="+ Mittelwert.ToString()+"Einheit“;  //In das Label mit der Eigenschaft Text wird eine Zahl  //geschrieben werden. Dies geht nur, wenn die Zahl vorher  //in einen „Text“ gewandelt wird.  Inhalte:   * Variablen (Typen, deklarieren, initialisieren) * Ausgabe * Zuweisung, Kommentare * = “Das Ergebnis ist“ + Variable; * Random * Alle speichern und wieder öffnen |