|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Properties and Indexer:** | | **Delegates:** 🡪 ähnlich wie „function pointers“ in C | | |
| **GUI-Begriffe 🡪 Form**: selbständiges, vollwertiges verschiebbares Fenster (System.Windows.Forms) **Control:** von .Net definierte GUI-Komponente (müssen in eine andere Control z.B. Panel/Form platziert werden **UserControl:** Basisklasse für selbst generierte GUI-Komponenten **Controls:** Variable innerhalb eines Control, mit der eine Lister der eingebetteten Komponenten vom Type Control verwaltet werden. Nur in der Liste enthaltene Komponenten sind sichtbar. **Size/ClientSize:** Grösse mit/ohne Rand u. Titelleiste. Menüleiste wird nicht berücksichtigt. **Modal-Eigenschaft:** Im Modal-Modus geöffnete Forms müssen geschlossen werden, bevor Eingaben für andere, darunterliegende Fenster möglich sind (Dialogfenster). | |
| **Events (Handling):** |  | | **API Entwicklung für Spezialgeräte**  Die **RobotCtrl-Library** kontrollliert den Robot (Motor, Switches, LED’s,…) | |
|  | |
| **GUI-Programmierung 1**. Eine **leere Solution** wird erstellt. **2.** Ein **„Windows – Form“ Projekt** wird der Solution hinzugefügt **3.** in der Klasse „Form1.cs“ wird nun ein **leeres GUI – Fenster** angezeigt. **4.** Mittels der **Toolbox** können nun verschiedene Elemente mittels Drag – and – Drop ins Fenster gezogen werden. **5.** Diese **Elemente** können nun mithilfe der **Properties** nach Wunsch angepasst werden. **6.** Wird auf ein Element doppelgeklickt, öffnet sich automatisch die Controller – Klasse und ein **Event – Handler** wird erstellt, in welchem implementiert werden kann, was bei dieser Aktion geschehen soll | | | |  |

|  |  |
| --- | --- |
| **Parameters**    The **out** keyword causes arguments to be passed by reference. It is like the **ref** keyword, except that **ref** requires that the variable be initialized before it is passed. To use an **out** parameter, both the method definition and the calling method must explicitly use the **out** keyword. | **Bitmaskierung Einlesen und Schreiben an Ports** |
|  |
| **Motorenansteuerung** Robot besitzt zwei IC LM629 über diese wird v und a eingestellt und Distanz gelesen. Zur Ansteuerung wird **MotorCtrl** erstellt, davon zwei Instanzen für Motor (l,r) erzeugt. **DriveCtrl** Klasse ist dafür zuständig H-Bridge ein-/auszuschalten. |
| **Robot-Fahrsteuerung Ziel:** Beide Motoren separat ansteuern (unabhängig, gleichzeitig und mit Hilfe Prozess-Steuerung -> überprüft stetig den Zustand des Prozesses und macht Anpassungen) **Steuerungsloop:** (Endlosschleife oder mit Timer) 🡪 Bei Endlosschleife minimales Sleep benötigt, damit übrige Softwareteile auch CPU-Leistung erhalten. Einzelne Durchläufe nicht gleiche Dauer.   1. Prozess-Abbild: Beinhaltet alle messbaren und beobachtbaren Sensor Daten 2. Berechnungen/Entscheidungen passieren auf Basis von Punkt 1 3. Neue Prozess-Parameter sind am Zyklusende gesamthaft und gleichzeitig zu setzen bzw. zu aktivieren.   **Common Run Parameters:** UserControl zur Form hinzufügen, Events SpeedChanged und AccelerationChanged abonnieren. Im EventHandler wird v und a in den TrackLine, TrackTurn und TrackArcRight/Left-UserControls gesetzt. |

|  |  |
| --- | --- |
| **Multithreading**  **New** 🡪Das Thread-Objekt ist erzeugt, aber noch nicht gestartet **Ready** 🡪Der Thread ist gestartet, lokal Speicher (stack) ist zugeteilt, er wartet nur noch auf die Zuweisung des Prozessors. **Running** 🡪 Der Thread führt seine Anweisungen auf dem Prozessor aus  **Stopped** 🡪 Der Thread existiert nicht mehr. Das Thread – Objekt jedoch schon und kann, falls referenziert, benutzt werden. Sonst 🡪 Garbage Collector  **Abort (requested)** 🡪 Ein geblockter Thread kann wieder in den ready- Zustand versetzt werden. **Blocked** 🡪 Der Thread muss warten, bis eine Dedingung erfüllt wird z.B **1.** Warten eines Timeouts oder auf das Ende eines anderen Threads **2.** Aufgerufene Betriebssystemroutine muss beendet werden **3.** Exception (z.B. von nicht vorhandenen File lesen) **Objects lock-pool** 🡪 C# bietet die Möglichkeit von Monitoren und Locks. Wenn ein Code- Block gelockt ist, kann nur eine bestimmte Anzahl Objekte darin arbeiten. Will ein weiteres Objekt darauf zugreifen, muss es warten, bis ein Lock wieder freigegeben wird. Die freien Locks werden im lock-pool gespeichert. **Objects wait-pool** 🡪 Will ein Objekt auf einen gelockten Codeblock zugreifen, wird es zuerst in den wait-pool verschoben. Sobald ein Lock frei wird, kann dieses im lock – pool „gewonnen“ werden (es gibt keine Reihenfolge, welcher Thread das Lock erhält) und somit auf den geschützten Code – Bereich zugreifen. Der wait-pool ist somit quasi das Wartezimmer für gelockte Code - Segmente |  |
| **Bedarf an Synchronisation bei Nebenläufigkeit** (Wenn parallel arbeitende Programmteile auf eine gemeinsame Ressource zurgreifen wollen) **Unterschied Semaphor und Mutex** (der Zähler Mutex kann nur die Werte 0,1 annehmen und Semaphore nicht-negative Werte) **Speziell an Sempaphor und Mutex** (sind geschützte Instanzen(Zähler) deren Operationen atomar ausgeführt werden) **Einsatz** (Mutex: Schreibender Zugriff auf gemeinesame Ressourcen, Datenbank Speicher, I/O Einheiten; Semaphor: Lesender Zugriff auf gemeinsame Ressourcen die begrenzt sind. Produzenten/Konsumenten Aufgaben, Bounded Buffer, Message-Passing, Channels) **Prozess** (Zugriff auf eigene Ressource -> Speicher, Prozessoren, I/O) **Thread** (teilt sich Ressourcen mit anderen Threads) **Sicherheit bei nebenläufigen Programmen** (keine Verklemmung (Deadlock) durch gegenseitige Zugriffe (Interferenzen) in kritischen Bereichen) **Lebendigkeit** (Kein Livelock, jeder Programmteil erhält eine faire Chance ausgeführt zu werden. -> unconditionaly fair, weak fair and strong fair. |
| (1)Welches werden die Prozesse der Mehrprozess Lösung sein? (2)Wo sind Programmänderungen/Erweiterungen zu machen?   |  | | --- | | Klasse RaceHorse und Turf weden zu prozessen | | Klasse RaceHorse mit main-Methode ergänzen |   Wo/Wie stellen Sie die prozessübergreifende Synchronisation dar?   |  | | --- | | Das EventWaitHadle in der Klasse Latch ändern und Name eines systemweiten Synchronisationsereignisses definieren.  *EventWaitHandle signal = new EventWaitHandle(false, EventResetMode.ManualReset, «hslu.prgsy.latch»);* |   Der Start der Rennpferd-Threads war ein Problem, deshalb wurde Thread.Sleep(500); in Zeile 44 eingefügt. Wie wird das Problem in der Mehrprozess Lösung behoben?   |  | | --- | | Warten durch Console.ReadLine(); statt Sleep  Ein Array von EventWaitHandler. Jedes RaceHorse Objekt erhält ein Array-Element und ruft in der Latch Methode Acquire(Erwerb) das set des EventWaitHandlers auf. Turf wartet mit WaitAll auf alle Array-Elemente. |   Ist das Rennen nun gerechter, als bei der Einzelprozess Lösung?   |  | | --- | | Nein! Es gelten die gleichen Bedingungen wie bei der Thread Lösung. Das EventWaitHandle der Klasse Latch besitzt einen Object-Wait und Lock-Pool durhc die alle Signalisation geht. | |
| Welche Einschränkungen besitzt die Klasse SimpleCheckDate?   |  | | --- | | EchoSever ist blockierend, d.h. er kann nur immer einen Client zur gleichen Zeit bearbeiten. *TcPClient client = listen.AcceptTcpClient();*  EchoServer ist nicht skalierbar, nicht stabil und stützt bei einem Fehler ab. (Behandelt nur TCP-Sockets) Es können nur C# Objekte verarbeitet werden. |   Mit welchem Konstrukt könnte eine Einschränkung von SimpleCheckDate behoben werden?   |  | | --- | | Handler (blockierend), Executor (skalierbar) |   Unhandled Exception: System.Net.Sockets.SocketException: Normalerweise darf jede Socketadresse (Protokoll, Netzwerkadresse oder Anschluss) nur jeweils einmal verwendet werden.   |  | | --- | | Der Port ist bereits druch einen andern Server belegt. |   Wie und mit welchen Parametern starten Sie das Telnet?   |  | | --- | | telnet IP(oder localhost) 4711, im telnet mit «open» beginnnen |   Was müssen Sie im Telnet tun, damit Sie auch eine Antwort erhalten?   |  | | --- | | Irgendwas eingeben + CR (Enter) -> man erhält immer eine Antwort |   Wenn der Client ein Web Browser ist und Kontakt mit SimpleCheckDate aufnimmt. Wie sieht die Anfrage aus, was passiert dann und warum?   |  | | --- | | <http://localhost:4711/12.3.04>, String wird nicht als gültiges Datum erkannt. http Kommando wird als request Zeile eingelesen. |   Wenn Sie einen Client für den SimpleCheckDate Server schreiben müssen. Wie muss die Socket Verbindung auf der Client-Seite erzeugt werden?   |  | | --- | | new TcpClient(host, port); | |