

CPU System Simulation

نشاط ٩

Concepts: Virtual Function, operator Overloading, Template:

- قم بإعادة صياغة الـ (class : Panel) الذي تم إنشاؤه في نشاط ٣ بحيث تقوم بإضافة الدالة :
void message () وهي دالة تقوم بطباعة الجملة (Panel Message) فقط.
- قم بإعادة صياغة الـ (class : ALU) الذي تم إنشاؤه في النشاط السابق ٨ حيث قم بإضافة الدالة :
void message () وهي دالة تقوم بطباعة الجملة (ALU Message) فقط.
- قم بإنشاء برنامج صغير يقوم بعمل التالي داخل الـ (main :
(١) أنشئ مؤشر من نوع Panel وليكن panptr ، ثم أنشئ object من نوع (class : Panel)
من خلال العامل new جاعلا المؤشر panptr يؤشر عليه.
(٢) استدعي دالة : message () من خلال المؤشر panptr ولاحظ الناتج.
(٣) أنشئ object من نوع (class : ALU) من خلال العامل new جاعلا المؤشر panptr يؤشر عليه.
(٤) استدعي دالة : message () من خلال المؤشر panptr ولاحظ الناتج
- قم بإضافة بسيطة على ما سبق حيث قم بحيث قم بإضافة الطلمة المحجوزة virtual قبل تعريف الدالة : message () الخاصة بالـ (class : Panel)
- قم بإعادة تشغيل البرنامج الذي أعدته ولاحظ الناتج.
- ما هو الفرق في الحالتين (قبل وبعد إضافة الكلمة المحجوزة virtual) ؟
- في نفس البرنامج السابق قم بإنشاء 3-objects من نوع (class : ALU) بالأسماء التالية:
(aluSum, alu2, alu1).
- ثم قم بكتابة الجملة : aluSum = alu1+alu2 حيث هذه الجملة تقوم بجمع قيمة المتغير accumulator التابعة لـ (object : alu1) مع قيمة الـ accumulator التابعة لـ (object : alu2) وإرجاع الناتج في الـ accumulator لـ (object : aluSum).
- ثم قم بكتابة الجملة : cout<< aluSum; حيث هذه الجملة تقوم بطباعة قيمة المتغير accumulator لـ aluSum فقط على الشاشة.
- لاحظ كلتا الجملتين لا تعملان إلا بعد إنشاء operator overloading للعاملين (+ , <<) لذلك قم بإنشاء global operator overloading لكل من العامل + والعامل <<.