هندسة البرمجيات

الوحدة الأولى

مقدمة في هندسة البرمجيات

م. ماجد ظاهر

الفصل الدراسي الأول

2022 - 2023

محتويات الوحدة

- مفاهيم عامة في هندسة البرمجيات
- تطوير هندسة البرمجيات
- صفات البرمجيات الجيدة
- دورة حيات نظام البرمجيات ومراحل تطويره
- أساسيات البرمجة الكينونية
- مخططات انسياب التحكم

مفاهيم عامة في هندسة البرمجيات

مفاهيم عامة في هندسة البرمجيات

- **النظام**

هو مجموعة من العناصر والقواعد المترابطة التي تعمل معاً لأداء مهمة.
حيث من الممكن ان يكون النظام كبيراً ومعقداً يتراوح ما بين منظومة صاروخية معقدة ومتقدمة الى نظام صغير مثل أداء الوضوء.

- **هندسة البرمجيات**

هي عملية بناء نظام متعدد الأجزاء بوساطة عدد من المختصين.

مفاهيم عامة في هندسة البرمجيات

- البرمجة يمكن أن تكون عملا فرديا ، بينما هندسة البرمجيات عمل فريق.

- المبرمج وحده يكتب برنامجا كاملا بينما مهندس البرمجيات يعتني بجزء من نظام متكامل لينضم هذا الجزء إلى أجزاء أخرى يعتني بها مهندسو برمجيات آخرون.

تطور هندسة البرمجيات

تطور هندسة البرمجيات

مشكلات صناعة البرمجيات في أواخر الستينات:

- التكاليف الباهظة المصاحبة لتطوير البرمجيات
- الجهد الكبير المبذول
- عدم إمكانية حساب الفترة الزمنية التي يستغرقها المشروع

هذه الأسباب أدت بصناعة البرمجيات إلى:

- العجز عن تنفيذ المشروع في الوقت المحدد له
- العجز عن تقليل التكاليف الباهظة
- العجز عن إنتاج برمجيات ذات جودة عالية

تطور هندسة البرمجيات

عوامل تبني مفهوم هندسة البرمجيات :

- ارتفاع تكاليف صناعة البرمجيات مقارنة مع المكونات المادية للحاسوب
- الدور المتزايد لصيانة البرمجيات
- التقدم السريع في التقنيات المادية للحاسوب
- الطلب المتزايد على البرمجيات
- التقدم في أساليب بناء البرمجيات
- الطلب على البرمجيات الكبيرة والمعقدة

أسس تطوير هندسة البرمجيات

أولاً: المصداقية Reliability (درجة الاعتماد على البرمجيات)

- **درجة الاعتمادية** هي صلاحية البرمجيات في حالة تطبيقها في بيئة معينة ولفترة زمنية محددة.
- **الصلاحية** هي خلو البرمجيات من الاخطاء بحيث تطبق بثقة وعدم الخوف من نتائج خاطئة ضمان تشغيل البرمجيات باحتمال قليل جدا من التعرض للعطب او التوقف عن العمل.
- **عوامل تحقيق الاعتمادية:**

1. تطوير برمجيات خالية من الأخطاء.
2. تطوير برمجيات استثنائية (لا يتم توقيف العمل بل اللجوء إلى وسيلة أخرى).
3. الكشف عن الأخطاء.

أسس تطوير هندسة البرمجيات

ثانيا: سهولة القراءة Readability

- تجنب الاختصارات غير الواضحة في كتابة البرامج واستخدام أسماء تعكس المسميات

- **من أجل تحقيق قابلية القراءة :**

1. استخدام مفهوم البرامج الفرعية وكتابتها على شكل أجزاء متعددة
2. استخدام مفهوم برمجة الكيانات.

أسس تطوير هندسة البرمجيات

ثالثا: جودة البرمجيات Software Quality

- بناء برمجيات تؤدي المهمات المطلوبة للجهة المستفيدة منها بكفاءة عالية,وتلتزم بالمقاييس المتبعة,وتحقق خصائص البرمجيات المتعارف عليها بين مراكز صناعة البرمجيات.

- **يمكن استنتاج ما يلي من التعريف:**

1. المهام التي تؤديها البرمجيات هي الأساس في درجة الجودة.
2. اعتماد المقاييس العالمية في صناعة البرمجيات سيؤدي إلى تصميم برمجيات ذات مواصفات جيدة يمكن أن يعتمد عليها.
3. تحقيق الخصائص المعارف عليها (الصيانة الميسرة، المقدرة على التطوير والتحوير).

أسس تطوير هندسة البرمجيات

ثالثا: جودة البرمجيات Software Quality

العوامل المؤثرة في جودة البرمجيات:

1. القدرة على إدامة وصيانة البرمجيات.
2. القدرة على توافقها مع أنظمة حاسوب مختلفة.
3. القدرة على أداء وظائفها بكفاءة عالية.

النقل : هل تنفذ البرمجيات على جهاز حاسوب من نوع آخر؟

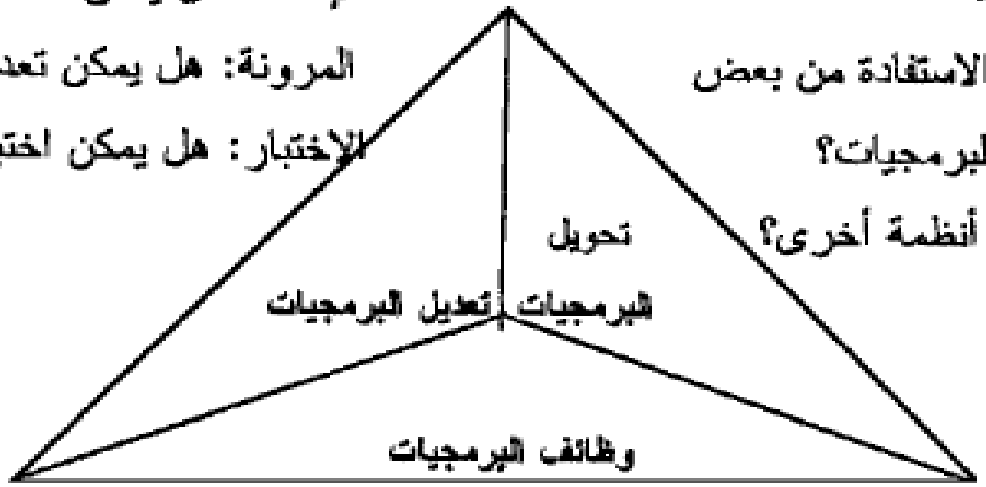
إعادة الاستعمال: هل يمكن الاستفادة من بعض مقاطع البرمجيات؟

الربط: هل يمكن ربطها مع أنظمة أخرى؟

الإدامة: هل يمكن تصحيح الخلل؟

المرونة: هل يمكن تعديله بسهولة؟

الاختبار: هل يمكن اختبار البرمجيات؟



صحة البرمجيات : هل تؤدي البرمجيات الوظيفة المطلوبة بصورة صحيحة؟

درجة الاعتمادية : هل تؤدي البرمجيات وظيفتها بصورة دقيقة في مرات للتنفيذ كافة؟

الكفاءة : هل تنفذ البرمجيات على أجهزة الحاسوب بالسرعة المطلوبة؟

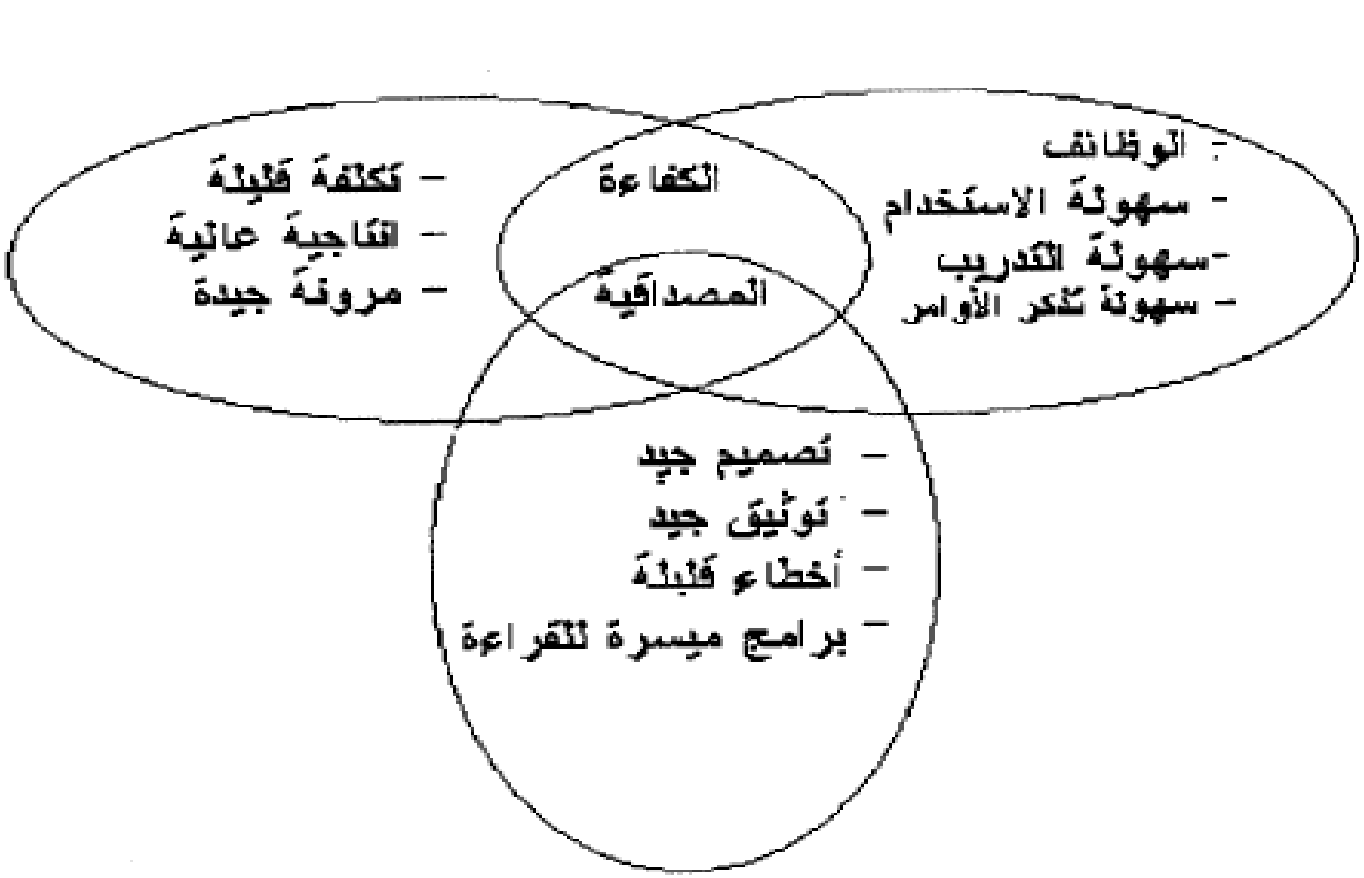
التكامل : هل البرمجيات متكاملة بحيث يمكن تمرير البيانات بين اجزائها ، مع منع غير المخولين من استغلالها؟

الشكل (1) : العوامل المؤثرة في جودة البرمجيات

- **مفهوم الجودة لدى الجهات ذات العلاقة**

- الهدف الرئيس من هندسة البرمجيات هو انتاج انظمة برمجيات ذات جودة عالية,ويختلف مفهوم الجودة من جهة اخرى,فعندما تبدأ مرحلة تشغيل نظام البرمجيات,فإن المفاهيم تتداخل بين ثلاث جهات:

- الجهة الأولى: الممولون الذين يتحملون تكاليف تطوير البرمجيات.
- الجهة الثانية: المستخدمون لهذه البرمجيات.
- الجهة الثالثة: فريق الصيانة الذي يقوم بتصحيح الأخطاء.



الشكل (2) : مفهوم الجودة لدى الجهات ذات العلاقة

صفات البرمجيات الجيدة

صفات البرمجيات الجيدة

1. البرمجيات الأقصر هي الأيسر متابعة.
2. تفضيل البرمجيات ذات القرارات الأقل.
3. تجنب تداخل القرارات.
4. التحديد الجيد لتركيب البيانات.
5. الإكثار من التوضيح والشرح.
6. الانسجام: تصميم فقرات البرمجيات بأسوب موحد ونمطي.
7. التكامل
8. الكفاءة: استخدام أقل حجم ممكن من الموارد (حجم التخزين، زمن المعالجة)
9. الفاعلية: قيام البرمجية بالوظائف المطلوبة وفقا لرغبة المستخدم

دورة حياة نظام البرمجيات ومراحل تطويره

دورة حياة نظام البرمجيات ومراحل تطويره

- **دورة حياة البرمجيات** هي مجموعة من الأنشطة التي تبدأ منذ بداية التفكير بالبرمجية,مروراً بتحليلها وتصميمها وإنتاجها واستخدامها.وتقسم الدورة إلى عدة **مراحل**:
  1. تحديد المتطلبات وتحليل النظام.
  2. التصميم.
  3. التحويل.
  4. الفحص والاختبار.
  5. التشغيل والتطبيق.
  6. التوثيق.
- مراحل إضافية : مرحلة وضع البرمجيات قيد التطبيق ، مرحلة الخروج من الخدمة.

المجموعات المؤثرة في تقدم البرمجيات وتطويرها:

- الإدارة
- محللو النظام
- المبرمجون ومبرمجو الأنظمة.
- مستخدمو النظام المقترح
- قسم الصيانة

مراحل تطوير حياة نظام البرمجيات

المرحلة الأولى : تحديد المسألة ومتطلبات المستفيد Requirements Phase

تهدف إلى دراسة احتياجات ومتطلبات المستخدمين وتحديدها, وبالتالي صياغة الأهداف والوظائف التي يجب ان يقوم بها النظام البرمجي المطلوب.

المرحلة الثانية: صياغة المتطلبات Specification Phase

دراسة المتطلبات التي تم جمعها من الجهة المستفيدة بهدف تبويبها وترتيبها لإحداث الربط والتسلسل بين مختلف أجزائها,ثم صياغة هذه المتطلبات إما بأسلوب لغوي, او باستخدام إحدى اللغات المناسبة لهذه المرحلة.

مراحل تطوير حياة نظام البرمجيات

المرحلة الثالثة: التخطيط Planning Phase

- تتطلب دراسة النظام القائم وتحليله وتحديد امكانية تعديله او بناء نظام جديد وقد يستلزم الامر حساب الجدوى الاقتصادية.

- يجب أن تبدأ الدراسة بما يلي:

1- تحديد المخرجات والمدخلات للنظام المقترح.

2-معالجة البيانات.

3-المعلومات او النتائج الخارجة من المعالجة.

مراحل تطوير حياة نظام البرمجيات

المرحلة الرابعة: تحليل النظام Analysis Phase

- تهدف إلى تقرير الحاجة إلى نظام جديد أو تعديل النظام السابق.
- يتضمن التحليل خطوتين هما:

1. تجميع الحقائق :ويمكن تجميع الحقائق من مصادر عدة مثل:
   - الأشكال المكتوبة.
   - الاستبيانات.
   - المقابلات.
   - المشاهدات.

2. تحليل الحقائق وفرزها وتصنيفها: ومن الوسائل المساعدة في ذلك:
   - المخططات الانسيابية System Flow Charts
   - جداول القرارات Decision Tables
   - انسيابية البيانات Data Flow Diagram

مراحل تطوير حياة نظام البرمجيات

المرحلة الخامسة: تصميم النظام Design Phase

- تتلخص خطوات تصميم النظام بما يلي:-
  - تحديد الأهداف.
  - تطوير تصميم نظام البرمجيات.
  - تحليل التكاليف والمنافع.
  - إعداد تقرير عن التصميم.
- يجب توضيح الأهداف : من ناحية الإدارة ، العمل ، التكاليف
- وخلال مرحلة تصميم النظام ينبغي دراسة:-
  - مستلزمات العمل والمدخلات والمخرجات.
  - الملفات والبيانات.
  - الرقابة والضبط على النظام.
  - الخطوات المتبعة في تنفيذ العمل (خطة العمل).
- لتحليل التكاليف والمنافع تعد قائمة بالتكاليف وقائمة بالمنافع.
- بعد الدراسة والتحليل يجب تلخيص النتائج وكتابة تقرير إلى الإدارة العليا

مراحل تطوير حياة نظام البرمجيات

المرحلة السادسة: تنفيذ النظام Implementation Phase

- يتم خلال هذه المرحلة بناء نظام المعلومات المقترح ويشمل:
  1. شراء المعدات أو استئجارها.
  2. كتابة البرامج وبناؤها.
  3. اختبار النظام.

- أهم الأنشطة التي تتضمنها :
  - كتابة البرامج وفحصها.
  - مرحلة التحويل المتوازي.

مراحل تطوير حياة نظام البرمجيات

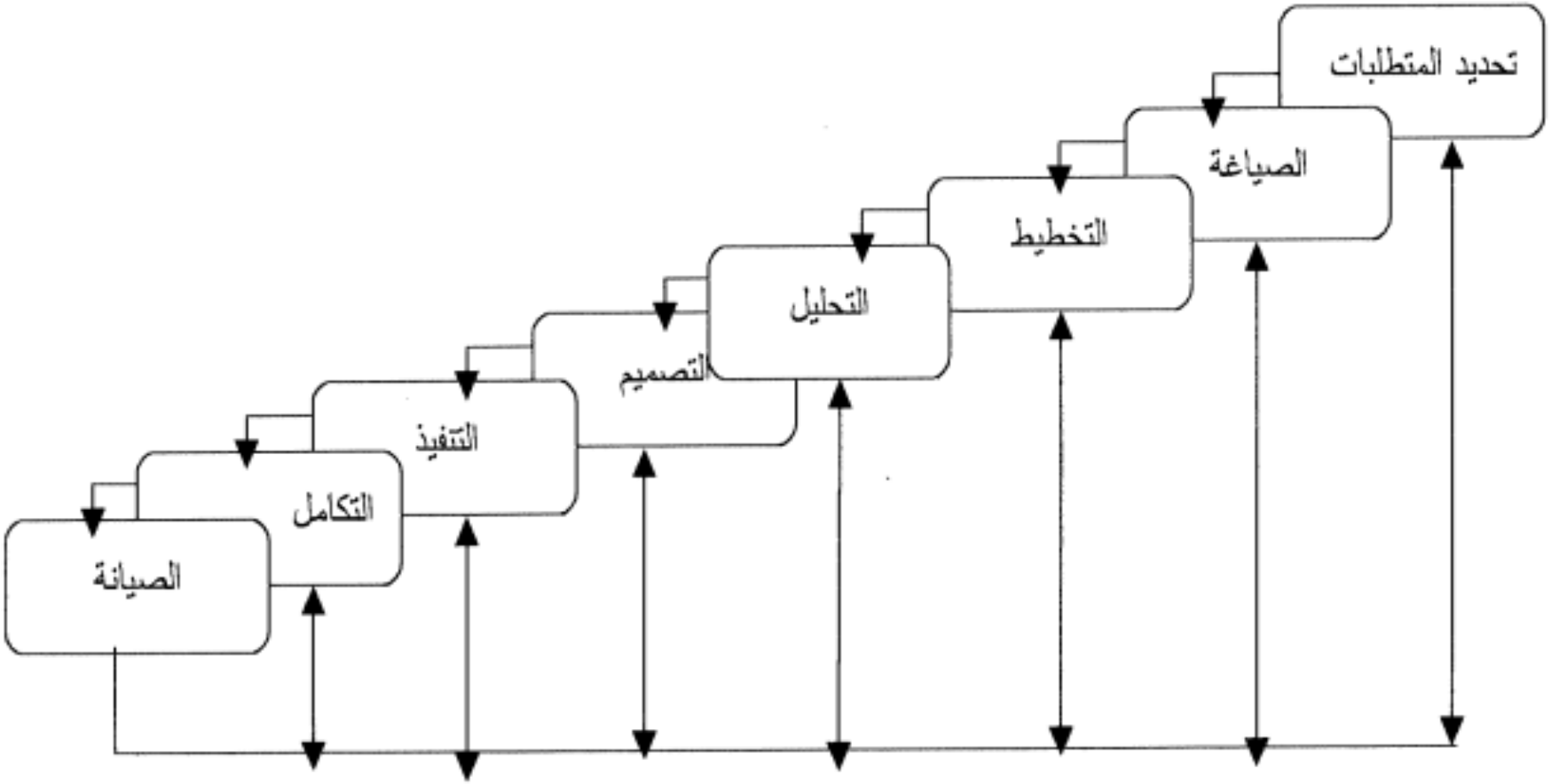
المرحلة السابعة: مرحلة التكامل Integration Phase

تعني جمع أجزاء النظام المختلفة لبناء النظام المتكامل(المنتج المطلوب)

المرحلة الثامنة: مرحلة الصيانة Maintenance Phase

تستمر هذه المرحلة مع النظام طوال دورة حياته,وتعد من أكثر المراحل تكلفة.

مراحل تطوير حياة نظام البرمجيات



الشكل (3) :الشلال المائي

أساسيات البرمجة الكينونية

البرمجة الكينونية
Object-Oriented Programming

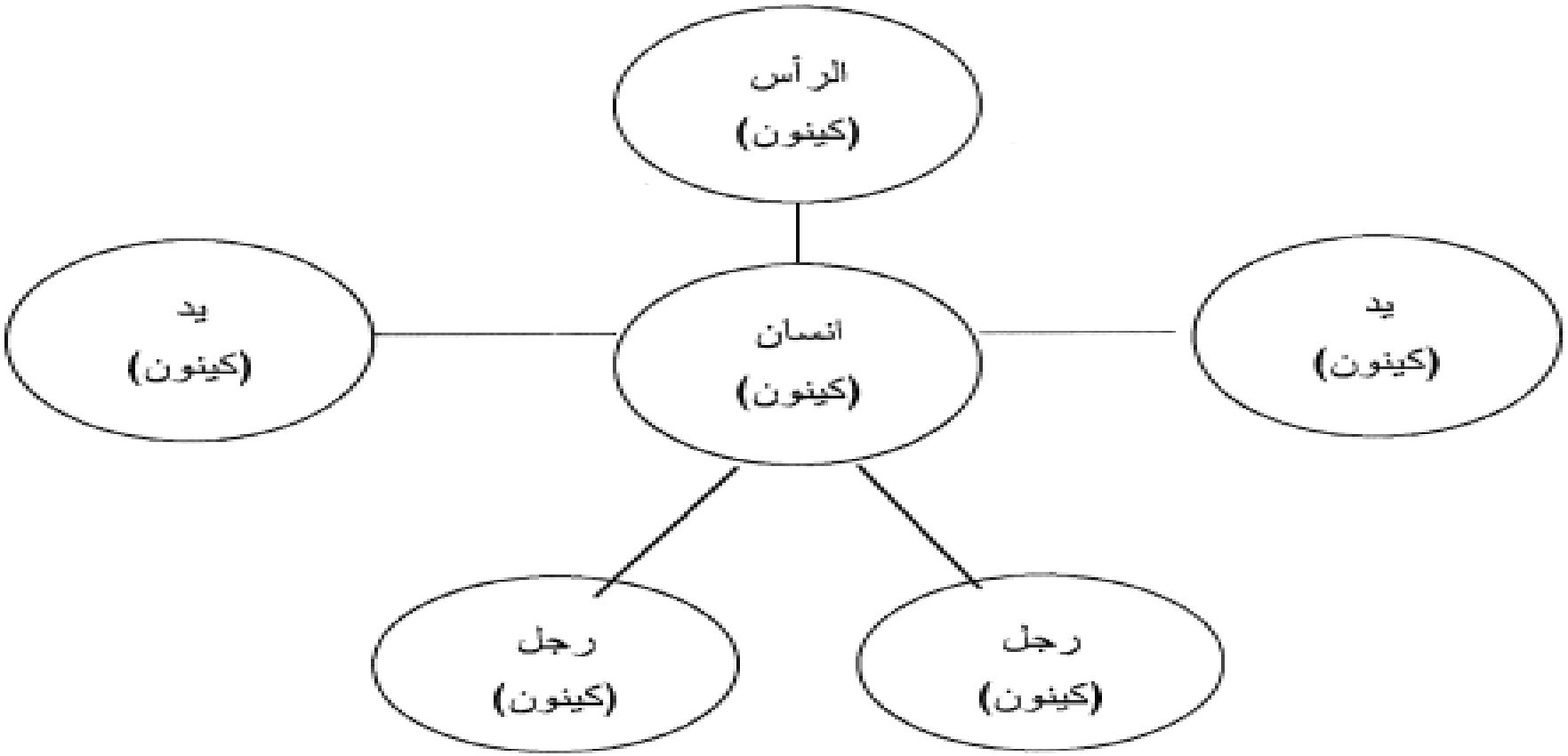
- تتجه في تصميم البرامج نحو تمثيل مسائل العالم الحقيقي باستخدام الكائنات والأصناف.
- توفر إمكانيات متعددة منها:التنظيم الجيد,والمقاطع البرمجية المتكاملة,والوضوح والانسجام,والكفاءة,والاستخدام المتعدد.
- تعد البرمجة الكينونية أسلوبا جديدا في التفكير وبناء البرامج بجمع حقول البيانات مع البرامج الفرعية التي تستند في عملها على تلك الحقول في بنية واحدة تسمى كينون Object

البرمجة الكينونية

الكينون Object: هو وحدة برمجية لها استقلاليتها وهو تجسيد لشئ واقعي يؤدي مهمة محددة، يحتوي بياناته الخاصة به كما يحتوي على العمليات التي يسمح بتنفيذها على تلك البيانات.

ومن اجل بناء برامج كينونية يستلزم الأمر اتباع أربع خطوات هي:

- تحديد تركيب الكينون المطلوب
- تحديد المتغيرات التي يتضمنها كل كينون وتعريف كل متغير
- تحديد البرامج الفرعية (الوسائل) المطلوبة لتنفيذ عمليات الكينون.
- تحديد أهداف كل برنامج فرعي وأهمية المتغيرات الأساسية وتحديد الشروط الاولية والنتائج المتوقعة.



شكل (4) : نموذج وكينونات

البرمجة الكينونية

- **الصف Class:** يستخدم لوصف مجموعة كينونات لها نفس السمات او الملامح, وتقوم بعمليات متشابهة. ومن الممكن ان يحتوي النظام المتكامل عدة صفوف وعادة ما يوجد ترابط بين كينونات تنتمي لصفوف مختلفة.
- مهام الكينونة هي بمثابة الخدمات التي تقدمها للكينونات الاخرى.
- يجب تحديد كينونات نظام البرمجيات وخصائصها والعمليات المصاحبة لها.
- ينظر إلى الأسماء الرئيسة باعتبارها كينونات والأفعال باعتبارها عمليات.

البرمجة الكينونية

مزايا البرمجة الكينونية:

1. خصوصية المعلومات وإمكانية اخفائها.
2. التوارث Inheritance: استخدام نفس السمات والملامح الموجودة لدى كينون بواسطة كينون اخر.حيث يمكن ان يرث كينون ما بعض سماته وخصائصه من كينون اخر.
3. الاستعمال المتعدد Polymorphism: استدعاء الكينون بمتغيرات مختلفة الانواع والحصول على نتائج مختلفة، حيث يستخدم برنامج فرعي بالاسم نفسه لأكثر من غرض.
4. استخدام الكينونات الديناميكية: حيث يتم تخصيص جزء من الذاكرة للكينونات خلال مرحلة تنفيذ البرامج,ويلغي التخصيص ويعاد حجم الذاكرة المقتطع الى وضعه الاصلي حال انتهاء دور الكينون.

مخططات انسياب التحكم

مخططات انسياب التحكم
Control Flow Graphs

- تصف هذه المخططات التركيب المنطقي لمقاطع البرمجيات,حيث المقطع هو عبارة عن وحدة برمجة من برامج (برنامج فرعي أو دالة)

- يتكون المخطط الذي يصف مقطعاً من رؤوس(Nodes) او دوائر صغيرة تمثل الجمل وحواف(Edges) تمثل انسياب التحكم بين الرؤوس.