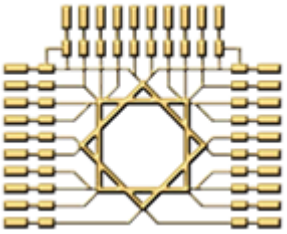
الجمهورية العربية السورية

المعهد العالي للعلوم التطبيقية والتكنولوجيا



الجلسة الأولى

البرمجة المتوازية - النياسب المتعددة

التمرين الأول: سباق اختراق الخزنة (Vault Hacking Race)

اكتب برنامج Java يحاكي سيناريو سباق بين قراصنة (hackers) يحاولون اختراق خزنة محمية بكلمة سر رقمية، بينما الشرطة في طريقها للقبض عليهم.

متطلبات البرنامج:

1. **صنف الخزنة (Vault)**:
   - تحتوي على كلمة سر رقمية عشوائية بين 0 و 9999
   - توفر دالة isCorrectPassword(int guess) للتحقق من التخمين
   - كل محاولة تخمين تستغرق 5 ميلي ثانية (لمحاكاة وقت المعالجة)
2. **نيسب القرصان التصاعدي (AscendingHackerThread)**:
   - يحاول تخمين كلمة السر بدءاً من 0 صعوداً إلى 9999
   - عند العثور على كلمة السر الصحيحة، يطبع رسالة ويُنهي البرنامج
3. **نيسب القرصان التنازلي (DescendingHackerThread)**:
   - يحاول تخمين كلمة السر بدءاً من 9999 نزولاً إلى 0
   - عند العثور على كلمة السر الصحيحة، يطبع رسالة ويُنهي البرنامج
4. **نيسب الشرطة (PoliceThread)**:
   - يعد تنازلياً من 10 إلى 0 (ثانية واحدة لكل عدد)
   - عند الوصول إلى 0، يطبع "Game over for you hackers" ويُنهي البرنامج
   - يمثل الوقت المتاح للقراصنة قبل وصول الشرطة
5. **البرنامج الرئيسي**:
   - يُنشئ خزنة بكلمة سر عشوائية
   - يُطلق النياسب الثلاثة في نفس الوقت
   - يستخدم أولوية Thread.MAX_PRIORITY لنياسب القراصنة

السلوك المتوقع

- إذا نجح أحد القراصنة في اختراق الخزنة قبل انتهاء الوقت، يفوز وينتهي البرنامج
- إذا انتهى الوقت (10 ثواني) قبل الاختراق، تفوز الشرطة وينتهي البرنامج

الأهداف التعليمية

1. فهم أساسيات البرمجة متعددة النياسب (Multithreading Basics)

- **الهدف**: تعلم كيفية إنشاء وإطلاق نياسب متعددة في Java
- **المهارات المكتسبة**:
  - استخدام Thread class وتوريثه
  - فهم دورة حياة النسيب (Thread lifecycle)

2. التعامل مع أولويات النياسب (Thread Priorities)

- **الهدف**: فهم كيفية تأثير أولويات النياسب على جدولة المعالج
- **المهارات المكتسبة**:
  - استخدام setPriority()
  - فهم Thread.MAX_PRIORITY و Thread.MIN_PRIORITY
  - ملاحظة أن الأولويات ليست ضماناً مطلقاً للتنفيذ

3. حالة السباق (Race Conditions)

- **الهدف**: فهم مفهوم التنافس بين النياسب
- **المهارات المكتسبة**:
  - ملاحظة أن النتيجة غير حتمية (non-deterministic)
  - فهم أن أي نيسب قد يفوز حسب:
    - قيمة كلمة السر العشوائية
    - جدولة نظام التشغيل
    - سرعة المعالج
  - إدراك الحاجة للتزامن (synchronization) في برامج أكثر تعقيداً

4. إدارة دورة حياة البرنامج (Program Lifecycle Management)

- **الهدف**: تعلم كيفية إنهاء البرنامج من نياسب مختلفة
- **المهارات المكتسبة**:
  - استخدام System.exit(0)
  - فهم أن أي نيسب يمكنه إنهاء البرنامج بالكامل
  - التعامل مع سيناريوهات انتهاء متعددة

5. التعامل مع المقاطعات (Thread Interruption)

- **الهدف**: فهم آلية sleep() ومعالجة الاستثناءات
- **المهارات المكتسبة**:
  - استخدام Thread.sleep()
  - معالجة InterruptedException
  - فهم أن sleep() قد يُقاطع

6. التصميم الكائني التوجه (OOP Design)

- **الهدف**: تطبيق مبادئ OOP في سياق البرمجة متعددة النياسب
- **المهارات المكتسبة**:
  - استخدام الأصناف المجردة (Abstract classes)
  - الوراثة (Inheritance) لمشاركة السلوك المشترك
  - التغليف (Encapsulation) في صنف Vault
  - استخدام Inner classes

7. التفكير المنطقي والتحليلي

- **الهدف**: تحليل سلوك البرنامج وتوقع النتائج
- **المهارات المكتسبة**:
  - حساب الوقت اللازم لكسر كلمات سر مختلفة
  - فهم أن كلمات السر القريبة من المنتصف (حوالي 5000) أصعب
  - تحليل احتمالية فوز كل نيسب

أسئلة للمناقشة

1. ما هي احتمالية فوز كل من القراصنة أو الشرطة؟
2. لماذا استخدمنا Thread.sleep(5) في دالة isCorrectPassword()؟
3. ماذا سيحدث إذا كانت كلمة السر هي 5000؟ من سيفوز؟
4. كيف يمكن تحسين فرص القراصنة في الفوز؟
5. ما هي المخاطر الأمنية لاستخدام System.exit() من نياسب مختلفة؟
6. لماذا استخدمنا Thread.MAX_PRIORITY للقراصنة؟ هل هذا يضمن فوزهم؟

تمارين إضافية (اختيارية)

1. أضف نيسب جديد يعبر عن قرصان ثالث يستخدم استراتيجية "Binary Search"
2. أضف عداد لعدد المحاولات التي قام بها كل قرصان
3. أضف واجهة رسومية لعرض تقدم كل قرصان
4. ما تراه مناسباً

يُطلب **استخدام *Git*** لبيان مراحل تطوير السؤال السابق.