Asignación de Prácticas Número 1

Programación Concurrente y de Tiempo Real

[1]Departamento de Ingeniería Informática
Universidad de Cádiz

PCTR, 2022

Contenido I

Objetivos de la Práctica

- Practicar las nociones elementales de programación en Java, en lo relativo a estructuras de control básicas, declaración y uso de variables, lectura de parámetros desde la línea de comandos, etc.
- Conocer las técnicas de generación de números aleatorios en Java y aplicarlas a la integración numérica.
- Practicar conceptos elementales de implementación de clases, de herencia y de implementación de interfaces, pues serán necesarios para la programación multihebra en Java.

Generación de Números Aleatorios en Java I

En Java es posible disponer de números aleatorios mediante tres técnicas diferentes:

- Utilizando el método `random()` de la clase `Math`
- Utilizando la clase `java.util.Random`
- Utilizando la clase `java.security.SecureRandom`
- Las dos primeras técnicas no se consideran criptográficamente seguras, pero la tercera sí.
- En este curso emplearemos la dos primera técnicas.

El Método `Math.random()` I

- Proporciona números pseudoaleatorios
- Los números se distribuyen de manera uniforme en el intervalo [0.0, 1.0)
- Si se necesitan números en rangos distintos, es necesario efectuar la transformación adecuada.
- Es un método **sincronizado**; diferentes hebras pueden compartir el mismo generador de forma segura...
- ... pero pagando un precio: la sincronización induce retardos.
- Si se desea evitar los retardos, hay que recurrir (haciéndolo adecuadamente) a la clase Random.

Utilizando el Método Math.random() I

```
1  public class metodoMathRandom {
2
3    public static void main(String[] args) {
4    double numAleatorio;
5      for (int i=0;i<100;i++){
6        numAleatorio=Math.random();
7        System.out.println(numAleatorio);
8      }
9
10   }
11 }
```

La Clase Random I

- Incluida en `java.util`
- Permite instanciar (como objetos) tantos generadores como se quiera, utilizando generadores de congruencias
- Es más adecuada cuando se desean múltiples flujos de datos aleatorios (parametrizando el constructor adecuadamente)
- Permite generar datos aleatorios de muy diversos tipos
- Si dos instancias se crean con la misma semilla, y se piden los mismos métodos (por ejemplo, desde hebras diferentes, la secuencia de número aleatorios obtenidos será la misma)
- No se recomienda su uso en aplicaciones que requieren seguridad criptográfica
- Ahora, navegamos al API de la clase...

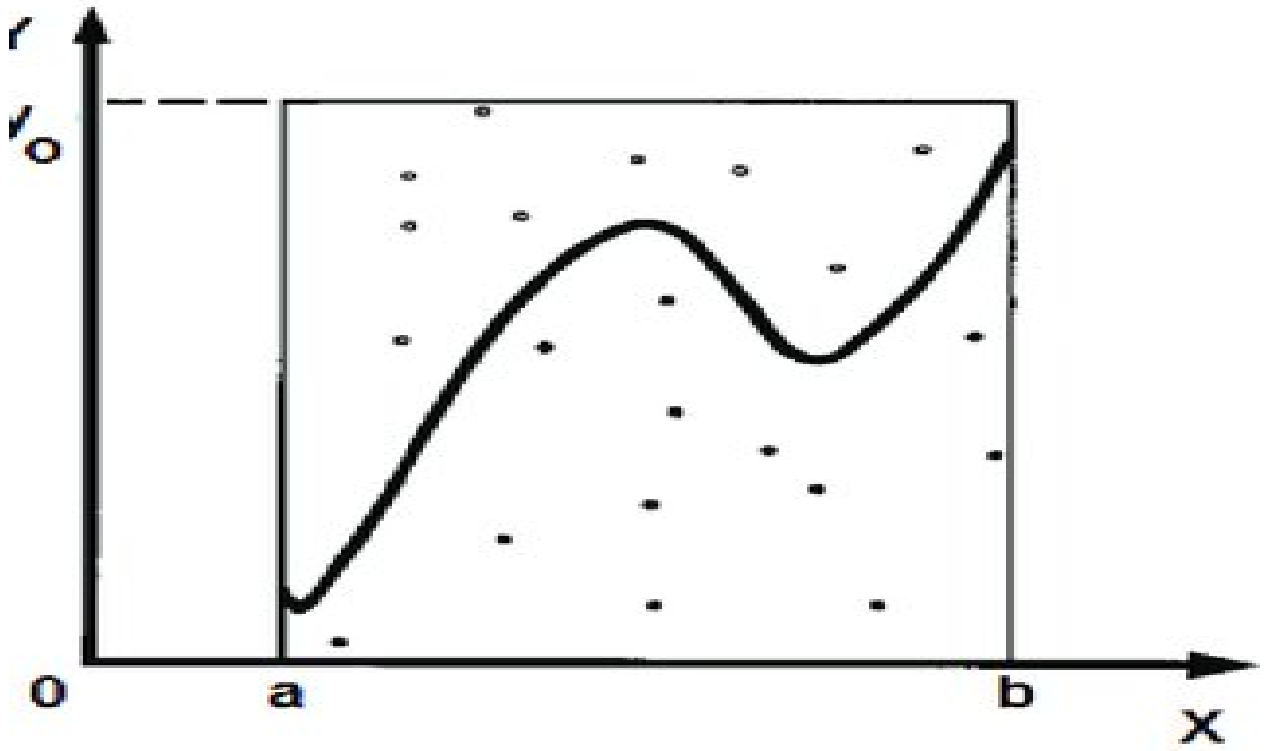
Utilizando La Clase Random() I

```
1  import java.util.Scanner;
2  import java.util.*;
3
4  public class nAleatorio {
5    public static void main(String[] args){
6      Scanner s = new Scanner(System.in);
7      int     n = s.nextInt();
8      Random  r = new Random();
9      for(int i=0; i<n; i++)
10       System.out.println(r.nextFloat());
11     for(int i=0; i<n; i++)
12       System.out.println(r.nextInt());
13     for(int i=0; i<n; i++)
           System.out.println(r.nextBoolean());
14     for(int i=0; i<n; i++){
15       System.out.println(r.nextGaussian());
16     }
17   }
18 }
```

Aplicando los Números Aleatorios: Integración Numérica de Monte-Carlo I

- Problema: calcular de forma aproximada la integral de una función en el intervalo $[0, 1]$
- Proponemos el método para funciones $f(x)$ tales que $\int_0^1 f(x) \leq 1$...
- ... aunque es fácilmente generalizable a intervalos y funciones más complejas
- El método lanza puntos aleatorios en el cuadrado de lado la unidad (cuya superficie es uno)
- Se cuentan aquellos puntos que caen debajo de la curva de la función
- La razón entre el número de puntos bajo la curva y el número total de puntos aproxima la integral
- La precisión aumenta cuando el número de puntos lanzados crece

Gráficamente...



El Algoritmo de Monte-Carlo para Integración Numérica

```
Procedimiento Monte-Carlo (n)
contador_exitos <- 0
Para i <- 0 Hasta n Con Paso 1 Hacer
  coordenada_x <- aleatorio (0,1)
  coordenada_y <- aleatorio (0,1)
Si coordenada_y <= f(coordenada_x)
  contador_exitos <- contador_exitos + 1
Fin Si
Fin Para
Escribir Integral aproximada: ,(contador_exitos/n)
Fin Procedimiento
```

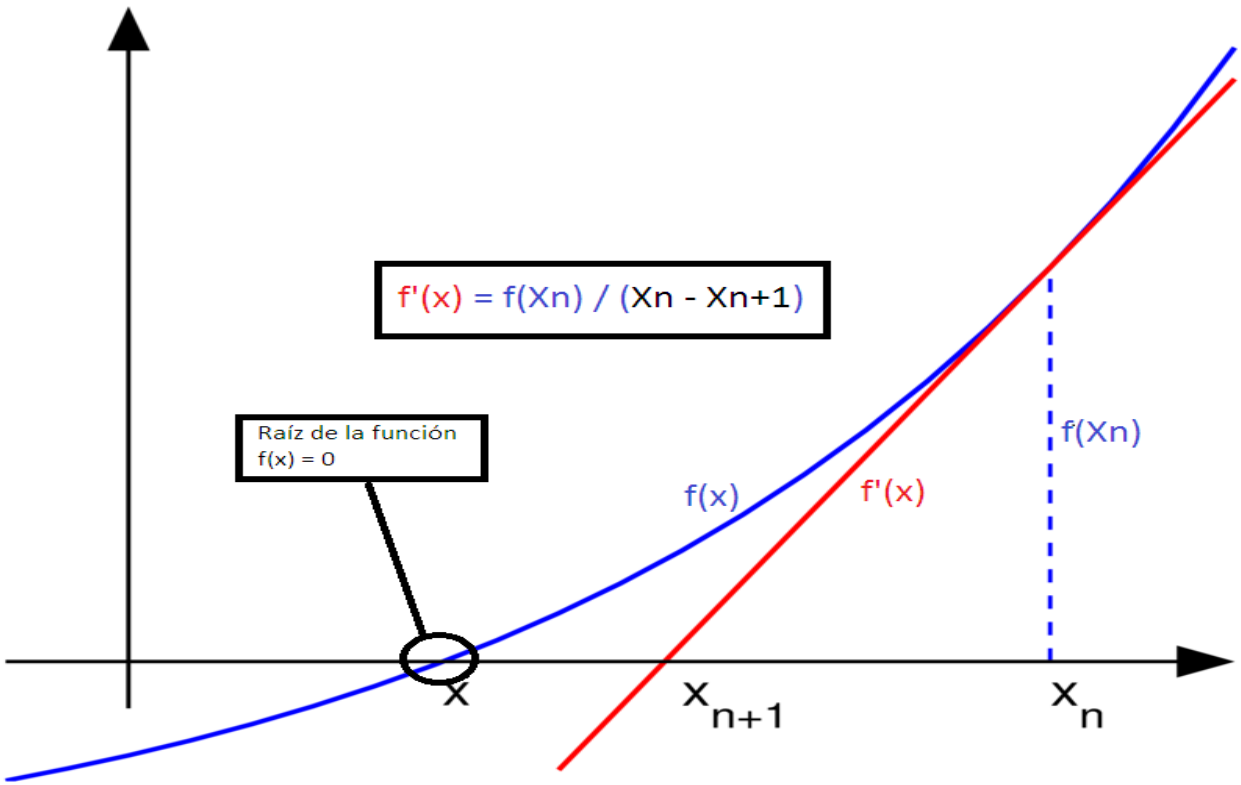
¿Y Ahora? Resolvemos el Ejercicio Número 3 I

- Aplicando el algoritmo anterior, aproximamos las integrales de las funciones $sin(x)$ y $f(x) = x$ en $[0, 1]$
- Comenzamos por la función $f(x) = x$ tiene una integral en el intervalo igual a 0,5
- Utilizamos el método `Math.random()` para generar coordenadas aleatorias...
- ... pero sugerimos que escriba una segunda versión utilizando la clase `Random`
- Dejamos 20 minutos para trabajar en esto...
- ... y seguimos

Ejercicio Número 2: Aproximando Raíces vía Newton-Raphson I

- En ocasiones encontrar el cero (la raíz) de una función en un intervalo dado no tiene solución analítica
- Existen múltiples técnicas numéricas que permiten resolver este problema de forma aproximada
- En esta ocasión, utilizaremos el método de Newton-Raphson
- Aplicable a funciones reales de variable real, continuas y derivables en el intervalo deseado
- Construye una sucesión de puntos a partir de uno dado inicialmente, que en el límite converge (o no) a la raíz de la función
- Es un método muy rápido en aproximar la raíz... pero ojo: a veces no converge

Gráficamente I



El Algoritmo de Newton-Raphson para Búsqueda de Raíces

```
Procedimiento Newton-Raphson (x0, iteraciones)
xN <- x0
Para i <- 0 Hasta iteraciones Con Paso 1 Hacer
  Si f'(xN) <> 0
    xN1 = xN - f(xN) / f'(xN)
    Escribir "Iteración: ", i, " Aproximación: ", xN1
  xN <- xN1
 Fin Si
Fin Para
Escribir "Resultado: ", xN
Fin Procedimiento
```

¿Y Ahora? Resolvemos el Ejercicio Número 2 I

- Aplicando el algoritmo anterior, aproximamos las raices de las funciones $cos(x) - x^3$ en $[0, 1]$ y $x^2 - 5$ en $[2, 3]$
- Comenzamos por la función $cos(x) - x^3$ que tiene la raíz aproximadamente en 0,8
- Dejamos 20 minutos para trabajar en esto...
- ... y seguimos

Transfiriendo Datos Desde La Línea de Comandos I

- En ocasiones resulta útil transferir los datos que un programa necesita desde la línea de comandos
- Para ello, Java proporciona como interfaz el array de cadenas que parametriza al método `main`
- De cada posición del array se extrae un parámetro que es una cadena de caracteres
- Si los datos que se necesitan son numéricos, es necesario efectuar una conversión de los mismos al tipo de datos adecuado

Otras Cuestiones a Trabajar en Casa Durante la Semana Próxima I

- Además de resolver y entregar los ejercicios de la asignación, es necesario trabajar durante la próxima semana en lo siguiente
- Modelo de herencia en Java: cómo funciona y cómo construir clases nuevas mediante herencia de clases ya existentes.
- Conviene practicar con ambos conceptos de forma intensiva, ya que las principales técnicas de multihebrado en Java hacen uso de herencia de clases y de implementación de interfaces.
- Finalmente, estudiar y probar el resto de código que se proporcionan en la carpeta de la práctica.