Programación Avanzada I. Práctica 5.1

Tema 6. Colecciones y Ordenaciones

Características de la práctica

En esta práctica se pretende que el alumno se familiarice con las colecciones (conjuntos y listas), así como con el orden natural y alternativo, que serán utilizados para crear conjuntos ordenados.

Ejercicio 1. Posicionamiento de páginas web (paquete rank)

Vamos a definir un proyecto que determine el posicionamiento de las páginas web de una red, atendiendo al número de clicks que se hacen en ellas y del posicionamiento de las páginas web que las enlazan. Para ello vamos a crear las clases `Site`, `Link` y `Web`, y posteriormente `SiteExtended` y `WebExtended`, todas en el paquete `rank`.

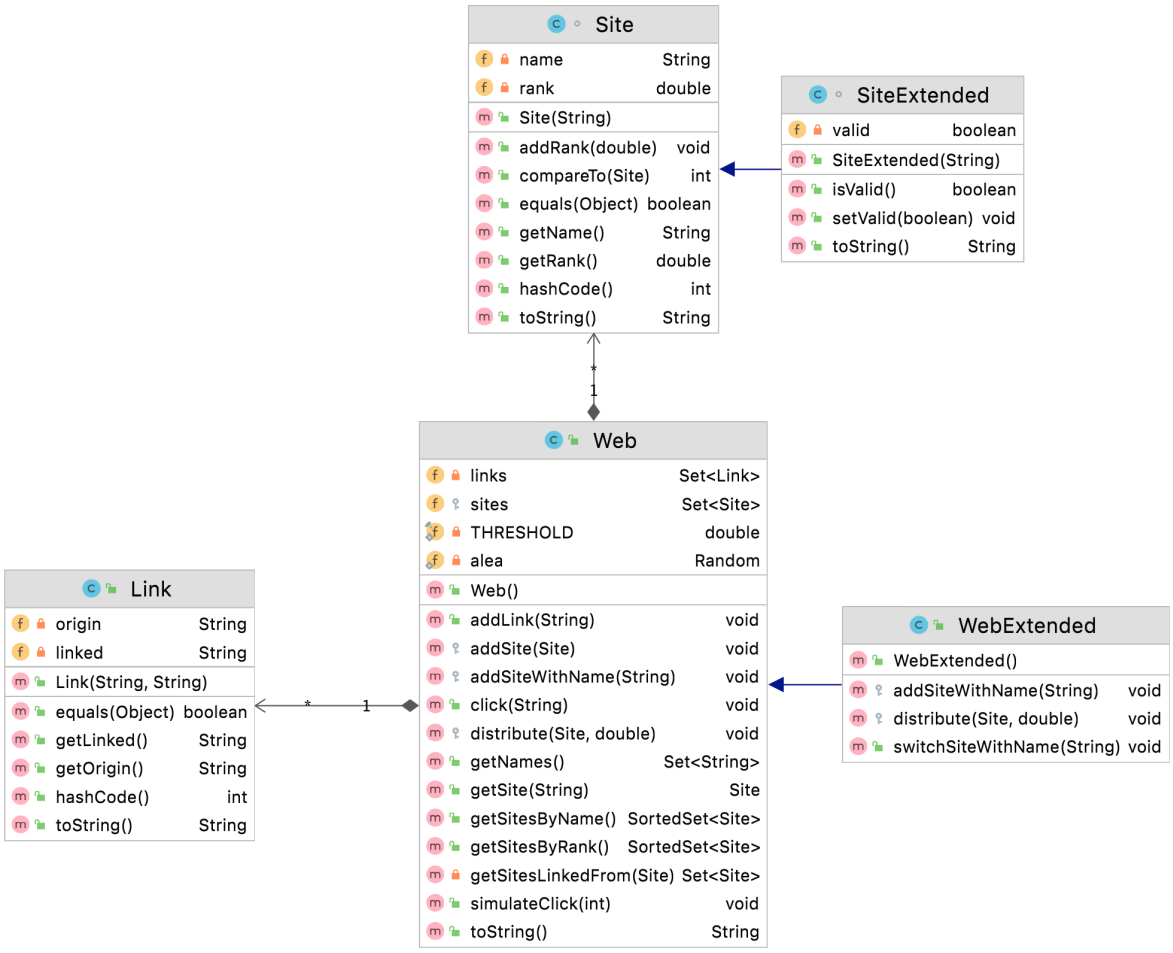

Figura 1: Diagrama de clases

La clase `Site`

Esta clase mantiene información de una página web. Por simplificar, solo mantendremos información del nombre de la página (de tipo `String`) y de un valor `double` que determina el posicionamiento de esta página (`double rank`). Cuanto mayor sea el valor `rank` mejor posicionada se considerará la página. Define para esta clase:

- Un constructor que crea una página, conocido el nombre:

  ```
  Site(String name);
  ```

  En este caso, el `rank`, será 0.
- Los getters para las dos variables de instancia.
- El método

  ```
  public void addRank(double r)
  ```

  que incrementa el `rank` de esta página sumándole el valor del argumento.
- Un criterio de igualdad, de forma que dos páginas son iguales si coinciden sus nombres sin tener en cuenta mayúsculas o minúsculas.
- Un orden natural para las páginas de manera que sea menor la que tenga un nombre lexicográficamente menor sin tener en cuenta mayúsculas y minúsculas.
- Una representación para los objetos de la clase de manera que una página de nombre `A`, con rango 5.25 se vea como `A(5.25000)`.

La clase `Link`

Esta clase mantiene información sobre un hipervínculo de una página a otra. Tendrá un `origin` (de tipo `String`) que será el nombre de una página que contiene el hipervínculo y un `linked` (de tipo `String`) que será el nombre de la página a la que se enlaza. Define para esta clase:

- Un constructor que crea un enlace conocidas la página origen y la enlazada:

  ```
  public Link(String org, String lnk);
  ```
- Los getters para las dos variables de instancias.
- Un criterio de igualdad, de forma que dos enlaces son iguales si coinciden su nombre de página origen y enlazada sin tener en cuenta mayúsculas o minúsculas.
- Una representación textual para los objetos de la clase de manera que un enlace de la página de nombre `A` a la página de nombre `B` se represente como `A->B`.

La clase `Web`

Esta clase mantiene información de todas las páginas que hay en la web y de todos los enlaces entre ellas. Las páginas las mantiene en un conjunto (`protected Set<Site> sites`) y los enlaces en otro conjunto (`private Set<Link> links`). Define para esta clase:

- Un constructor sin argumentos que inicialice correctamente las estructuras como conjuntos vacíos.
- El método

  ```
  protected void addSite(Site site)
  ```

  que añade `site` al conjunto de páginas.
- El método

  ```
  protected void addSiteWithName(String name)
  ```

  que crea un página con nombre `name` y la añade con el método `addSite`.
- El método

  ```
  public void addLink(String dataLink)
  ```

  que dado como argumento la cadena `dataLink` que debe tener la forma `A->B`, añade una página con nombre `A` y otra con nombre `B` al conjunto de páginas (con el método `addSiteWithName`), crea un enlace con estos nombres y lo añade al conjunto de enlaces. Este método puede fallar si la forma de `dataLink` no es la correcta. En ese caso lanza una `IllegalArgumentException` con un mensaje que incluya el `dataLink` que ha dado el problema.

- El método

  ```
  public Site getSite(String name)
  ```

  que devuelve la página de nombre `name` del conjunto `sites` sin diferenciar mayúsculas o minúsculas. Si no existe esa página se lanza una `NoSuchElementException` indicando tal circunstancia.

- El método

  ```
  public Set<String> getNames()
  ```

  que devuelve un conjunto con los nombres de las páginas de esta web.

- El método

  ```
  private Set<Site> getSitesLinkedFrom(Site pagina)
  ```

  que, dada una página, devuelve un conjunto con todas las páginas enlazadas desde ésta.

- El método

  ```
  protected void distribute(Site site, double prize)
  ```

  que distribuye el valor `prize` de la siguiente manera:

1. Si `prize` es menor que cierto umbral (declarado como `private static final double THRESHOLD = 1E-5` en la clase) no se hace nada. En otro caso se sigue con los siguientes pasos.
2. La mitad de `prize` se utiliza para incrementar el `rank` de `site`.
3. La otra mitad se distribuye equitativamente de forma **recursiva** entre las páginas enlazadas desde `site`. Así, si `site` tiene `n` enlaces a otras páginas, se distribuirá (de forma **recursiva**) a cada una de ellas el valor `prize/(2*n)`. Si `site` no tiene páginas enlazadas, el valor `prize/2` se pierde.

- El método

  ```
  public void click(String name)
  ```

  que, dado un nombre de página, distribuye (con el método `distribute`) a esa página el valor `1`. Si se produce un error porque la página no existe, no hace nada y se ignora el mensaje.

- El método

  ```
  public void simulateClick(int numClick)
  ```

  que repite `numClick` veces el proceso de seleccionar aleatoriamente una página y hacer `click` sobre ella. Si no hubiera ninguna página en la web no hace nada. Para obtener un valor aleatorio, crea una variable de clase `alea` de tipo `Random` en inicialízala con el valor `new Random(1)` (para poder predecir los resultados en las pruebas). Se debe crear una lista con todas las páginas y con ´alea´ obtener una posición aleatoria de esa lista para después simular el click en la página que se encuentra en esa posición.

- El método

  ```
  public SortedSet<Site> getSitesByName()
  ```

  que devuelve el conjunto de páginas ordenadas por el orden natural.

- El método

  ```
  public SortedSet<Site> getSitesByRank()
  ```

  que devuelve el conjunto de páginas ordenadas por el valor `rank` en orden decreciente y a igualdad de `rank`, por el orden natural.

- Una representación textual de la web, de manera que aparezca la palabra `Web` seguida del conjunto de páginas y del conjunto de enlaces entre paréntesis y separadas por coma (En el ejemplo siguiente, lo primero que aparece es la representación de una web).

La clase `MainRank`

Se proporciona la clase principal `MainRank` que crea una web, le añade unos enlaces, simula 4000 clicks y muestra los resultados (se han introducido saltos de línea por legibilidad).

La salida debería ser:

```
Web([A(0.00000), B(0.00000), C(0.00000), D(0.00000), E(0.00000), F(0.00000), G(0.00000),
H(0.00000), I(0.00000), J(0.00000)], [F->G, G->H, B->C, D->F, F->H, A->C, E->H, A->D,
B->F, J->C, I->C, E->B, G->E])
Paginas ordenadas alfabeticamente
[A(199.50000), B(266.47491), C(526.24142), D(251.87500), E(265.90868), F(382.55392),
G(295.63776), H(431.52135), I(200.00000), J(222.50000)]
Paginas ordenadas por rank
[C(526.24142), H(431.52135), F(382.55392), G(295.63776), B(266.47491), E(265.90868),
D(251.87500), J(222.50000), I(200.00000), A(199.50000)]
```

Ejercicio 2. Mejora del algoritmo

Algunos internautas, con objeto de mejorar el posicionamiento de sus páginas, crean cientos de páginas ficticias que enlacen con la suya. Las empresas de los buscadores intentan limitar este efecto y cuando detectan páginas ficticias las consideran no válidas para el cálculo del posicionamiento. Para implementar esta funcionalidad se van a crear las clases `SiteExtended` y `WebExtended` en el mismo paquete.

La clase `SiteExtended`

La clase `SiteExtended` extiende a la clase `Site` e incluye en su estado una variable `boolean valid` que indica si la página se debe tener en cuenta para el cálculo del posicionamiento. Para esta clase define:

- Un constructor que crea una página extendida conocido el nombre. La variable `valid` se inicializará a `true`.

  ```
  public SiteExtended(String name)
  ```

- El método

  ```
  public void setValid(boolean b)
  ```

  que cambia el valor de la variable `valid` al valor `b`.

- El método

  ```
  public boolean isValid()
  ```

  que devuelve el valor de la variable `valid`.

- Una representación de los objetos de la clase de manera que una página con nombre `A` y `rank` 5.25 que sea válida se mostrará como `A(5.25000)` y si no es válida como `A(5.25000)*`.

La clase `WebExtended`

Una `WebExtended` es una `Web` pero que ***solo*** permite incluir páginas extendidas. A la hora de calcular el posicionamiento de una página, no distribuirá a aquellas que no sean válidas. Para esta clase define:

- El método

  ```
  protected void addSiteWithName(String name)   @Redefinicion
  ```

  que crea una página extendida con nombre `name`, y llama al método `addSite` para añadirla al conjunto de páginas.

  NOTA Este es el único método en la clase que permite añadir páginas al conjunto de páginas por lo que asegura que todas las que se añadirán serán páginas extendidas.

- El método

  ```
  protected void distribute(Site site, double prize)   @Redefinicion
  ```

  de manera que si la página `site` no es válida, no hace nada. En otro caso, se comporta como lo hacía la superclase.

  NOTA Dado que el argumento `site` es de tipo `Site`, será necesario hacerle un cast para convertirlo en `SiteExtended` y así poder conocer si es válida, `SiteExtended siteEx = (SiteExtended)site`. El cast es seguro debido a la nota anterior.

- El método

  ```
  protected void switchSiteWithName(String name)
  ```

  que busca la página con nombre `name` y cambia el estado de la página válida a inválida o viceversa.

  NOTA Ten en cuenta la nota incluida en el método `distribute`.

- Modificar la clase `MainRank` para que ahora cree una `WebExtended` en lugar de una `Web` y antes de mostrar nada, invalida las páginas `A`, `I` y `J`.

La salida debería ser:

```
Web([A(199.50000), B(266.47491), C(526.24142), D(251.87500), E(265.90868),
F(382.55392), G(295.63776), H(431.52135), I(200.00000), J(222.50000)],[B->C,
F->G, G->H, A->C, D->F, F->H, A->D, E->H, B->F, J->C, I->C, E->B, G->E])
Páginas ordenadas alfabéticamente
[A(0.00000)*, B(270.60286), C(254.14847), D(194.00000), E(278.41455),
F(348.64847), G(297.66142), H(430.17883), I(0.00000)*, J(0.00000)*]
Páginas ordenadas por rank
[H(430.17883), F(348.64847), G(297.66142), E(278.41455), B(270.60286),
 C(254.14847), D(194.00000), A(0.00000)*, I(0.00000)*, J(0.00000)*]
```