Zestaw 7

Język Java

Termin 19.12.2018

Klasy umieścić w pakiecie **serialize**.
Dodać komentarze i użyć narzędzia **javadoc** do wygenerowania dokumentacji.
Szczegóły dotyczące serializacji można znaleźć w wykładzie.

Zadanie 1. `Circular`

Napisać klasę `Circular` implementującą bufor cykliczny. Elementami bufora są obiekty klasy implementującej interfejs `Serializable`. Klasa ma być klasą *generic* i implementować interfejsy `java.io.Serializable` oraz `java.util.concurrent.BlockingQueue<E>`, zatem jej deklaracja powinna wyglądać w następujący sposób:

```
public class Circular<E extends Serializable>
  implements Serializable, BlockingQueue<E>
```

Klasa ma wykorzystywać implementacje tablicową bufora cyklicznego. W tym celu należy stworzyć pole prywatne `E[] buf`. Aby stworzyć tablicę typu zmiennego można użyć kodu

```
    buf = (E[]) new Serializable[size];
```

Proszę zapoznać się z tzw. problemem *generic array creation*. Ponieważ klasa implementuje interfejs `BlockingQueue<E>`, implementuje także `Collection<E>`, `Iterable<E>`, `Queue<E>`. Stworzyć konstruktor, który jako argument przyjmuje wielkość bufora. W szczególności należy zaimplementować metody `add`, **`offer`**, `put` oraz `remove`, **`poll`**, `take` (przez pogrubione metody można wyrazić pozostałe by nie powielać kodu). Proszę zapoznać się z ich działaniem w API powyższych interfejsów. Metody te powinny mieć modyfikator `synchronized` oraz wywoływać operacje `notify()` i `wait()` (dotyczy `put` i `take`) na `this`.

Aby zapobiec *wyciekowi pamięci*, po usunięciu obiektu z bufora należy przypisać odpowiedniemu elementowi tablicy wartość `null`.

Podczas *serializacji/deserializacji* bufora cyklicznego należy zapisać/odczytać jedynie używane, a nie wszystkie elementy `buf`. Dlatego należy zaimplementować własne metody `writeObject(ObjectOutputStream)` oraz `readObject(ObjectInputStream)`.

Proszę zaimplementować metodę `public <T> T[] toArray(T[] a)`. Aby stworzyć (gdy jest to konieczne) nową tablicę typu `T[]`, można użyć kodu

```
java.lang.reflect.Array.newInstance(a.getClass().getComponentType(), len);
```

Proszę zapoznać się z pakietem `reflect`.

Proszę napisać metodę `toString()`. Dla każdego elementu tablicy `buf` należy wypisać indeks w tej tablicy, odległość od głowy bufora, typ obiektu i napis skojarzony z elementem, lub “Empty” gdy element tablicy nie przechowuje referencji do żadnego obiektu.

Napisać programy, które

1. dodają elementy typu `String` do bufora i zapisują (serializują) go do pliku (z rozszerzeniem `.ser`)
2. wczytują bufor cykliczny z pliku (deserializują) i wypisują jego zawartość (`toString`)

Zadanie 2. `CircularIterator`

Napisać klasę `CircularIterator<E>` realizującą interfejs `java.util.Iterator<E>`. Obiekt tej klasy ma być zwracany przez metodę `iterator()` bufora. **Sprawdzić** poprawność implementacji wykonując pętlę *for-each* na buforze:

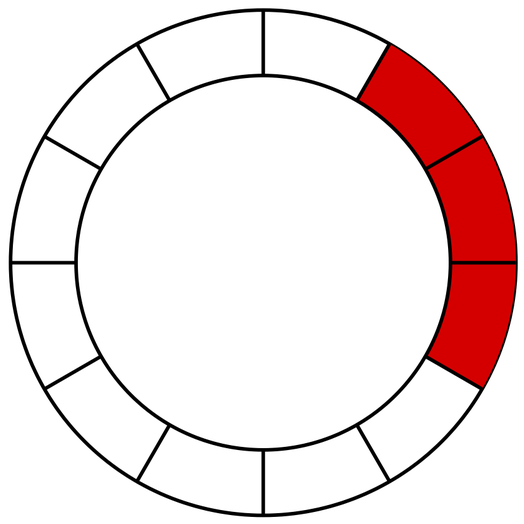
```
Circular<String> circular = new Circular<String>(16);
// Dodać elementy do bufora
for(String s : circular) {...}
```

Opcjonalnie:

> Uwaga wykonanie operacji `remove()` usuwa ostatnio zwrócony przez iterator element z **bufora**. Gdy dokonano modyfikacji bufora podczas użycia iteratora należy wyrzucić wyjątek `ConcurrentModificationException`.

Zadanie 3. `CircularPanel` (dodatkowe)

Klasa `CircularPanel` ma być graficzną nakładką na klasę `Circular` i dziedziczyć po klasie `java.swing.JPanel`. Przykładowa reprezentacja bufora zawierającego 3 elementy o maksymalnej pojemności 12 elementów, może wyglądać następująco:



W funkcji `main` należy stworzyć odpowiednie okienko wyświetlające panel oraz uruchomić osobne wątki zapisujące i odczytujące do/z bufora - proszę napisać osobne klasy `Putter` i `Getter` dziedziczące po`Thread` i wyświetlające informacje także na konsole. Zademonstrować poprawność operacji blokujących.

> Do uśpienia wątku na pewien czas należy użyć metody `sleep()`.

Obiekt `CircularPanel` powinien odświeżać obraz co jakiś czas (zaimplementować interfejs `Runnable`) lub po każdej operacji na buforze (wówczas należy to uwzględnić w klasie `Circular`).

Przed zakończeniem działania programu wypisać napis zwrócony przez `toString()` dla bufora.

Andrzej Görlich
agoerlich@netmail.if.uj.edu.pl
http://th.if.uj.edu.pl/~atg/Java

Print: Portrait, A4, Black and white, margins: none