Java 2 Notes

27 August 2024 10:22

Sure! Below are some numerical or visual examples for each of the topics:

1. **Enums**:
   - Example: `enum Day { SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY }`
   - Visual: Imagine a week calendar with fixed days (Sunday to Saturday) that cannot change.

2. **Collections**:
   - Numerical Example: `List<Integer> numbers = new ArrayList<>(Arrays.asList(1, 2, 3, 4, 5));`
   - Visual: Think of a shopping cart where you can add and remove items, or a container holding different shapes.

3. **ArrayList & Iterator**:
   - Example:
     ```java
     ArrayList<String> fruits = new ArrayList<>();
     fruits.add("Apple");
     fruits.add("Banana");
     Iterator<String> it = fruits.iterator();
     while (it.hasNext()) {
         System.out.println(it.next());
     }
     ```
   - Visual: A list where you can add or remove items and iterate through them one by one.

4. **Stack**:
   - Example:
     ```java
     Stack<Integer> stack = new Stack<>();
     stack.push(10);
     stack.push(20);
     System.out.println(stack.pop()); // Outputs 20
     ```
   - Visual: A stack of plates; you can only add or remove the top plate.

5. **Queue & LinkedList**:
   - Example:
     ```java
     Queue<String> queue = new LinkedList<>();
     queue.add("First");
     queue.add("Second");
     System.out.println(queue.poll()); // Outputs "First"
     ```
   - Visual: A line of people waiting for a service; first person in line gets served first.

6. **PriorityQueue**:
   - Example:
     ```java
     PriorityQueue<Integer> pq = new PriorityQueue<>();
     pq.add(3);
     pq.add(1);
     ```

```
    pq.add(2);
    System.out.println(pq.poll()); // Outputs 1
```
   - Visual: A hospital waiting room where patients with higher priority (e.g., critical cases) are treated first.

7. **ArrayDeque**:
   - Example:
     ```java
     Deque<Integer> deque = new ArrayDeque<>();
     deque.addFirst(1);
     deque.addLast(2);
     System.out.println(deque.removeFirst()); // Outputs 1
     ```
   - Visual: A double-ended queue where you can add or remove items from both ends.

8. **HashSet**:
   - Example:
     ```java
     HashSet<String> set = new HashSet<>();
     set.add("Apple");
     set.add("Banana");
     set.add("Apple"); // Duplicate, won't be added
     System.out.println(set); // Outputs [Apple, Banana]
     ```
   - Visual: A basket where duplicates are not allowed.

9. **LinkedHashSet**:
   - Example:
     ```java
     LinkedHashSet<String> linkedSet = new LinkedHashSet<>();
     linkedSet.add("Apple");
     linkedSet.add("Banana");
     linkedSet.add("Orange");
     System.out.println(linkedSet); // Outputs [Apple, Banana, Orange] in insertion order
     ```
   - Visual: A notebook where entries are unique and ordered by when you wrote them.

10. **TreeSet**:
    - Example:
      ```java
      TreeSet<Integer> treeSet = new TreeSet<>(Arrays.asList(5, 1, 3));
      System.out.println(treeSet); // Outputs [1, 3, 5] in sorted order
      ```
    - Visual: A sorted list of items, like alphabetical ordering in a phonebook.

11. **HashMap**:
    - Example:
      ```java
      HashMap<String, Integer> map = new HashMap<>();
      map.put("Apple", 10);
      map.put("Banana", 20);
      System.out.println(map.get("Apple")); // Outputs 10
      ```
    - Visual: A dictionary where each word (key) has a meaning (value).

12. **TreeMap**:

  - Example:
    ```java
    TreeMap<String, Integer> treeMap = new TreeMap<>();
    treeMap.put("B", 2);
    treeMap.put("A", 1);
    System.out.println(treeMap); // Outputs {A=1, B=2} in sorted order
    ```
  - Visual: A phonebook sorted by names.

13. **Array Class**:
  - Example:
    ```java
    int[] arr = {5, 2, 8};
    Arrays.sort(arr);
    System.out.println(Arrays.toString(arr)); // Outputs [2, 5, 8]
    ```
  - Visual: A line of numbers being rearranged into ascending order.

14. **Collection Class**:
  - Example:
    ```java
    List<String> list = Arrays.asList("apple", "banana", "cherry");
    Collections.sort(list);
    System.out.println(list); // Outputs [apple, banana, cherry]
    ```
  - Visual: Organizing a set of cards in alphabetical order.

15. **Thread**:
  - Example:
    ```java
    Thread t = new Thread(() -> System.out.println("Thread running"));
    t.start(); // Runs in parallel
    ```
  - Visual: Two people working on different tasks simultaneously.

16. **Runnable**:
  - Example:
    ```java
    Runnable task = () -> System.out.println("Task running");
    new Thread(task).start();
    ```
  - Visual: A blueprint for a task that can be executed by a worker (thread).

17. **Race Condition**:
  - Example:
    ```java
    int count = 0;
    Thread t1 = new Thread(() -> { for(int i = 0; i < 1000; i++) count++; });
    Thread t2 = new Thread(() -> { for(int i = 0; i < 1000; i++) count++; });
    t1.start();
    t2.start();
    ```
  - Visual: Two people trying to write to the same chalkboard at the same time, leading to unpredictable results.

18. **Thread State**:
  - Example:

```java
Thread t = new Thread(() -> {});
System.out.println(t.getState()); // Outputs NEW
t.start();
System.out.println(t.getState()); // Outputs RUNNABLE
```
- Visual: Different phases of a task, from not started (NEW) to actively running (RUNNABLE).

19. **Class Loaders**:
  - Example:
```java
ClassLoader classLoader = MyClass.class.getClassLoader();
System.out.println(classLoader);
```
  - Visual: A librarian fetching books (classes) from the library (JVM).

20. **Streams**:
  - Example:
```java
List<Integer> numbers = Arrays.asList(1, 2, 3, 4);
int sum = numbers.stream().filter(n -> n % 2 == 0).reduce(0, Integer::sum);
System.out.println(sum); // Outputs 6 (sum of even numbers)
```
  - Visual: A water stream where only certain objects pass through a filter, and the result is combined.

21. **Annotations**:
  - Example:
```java
@Override
public String toString() { return "Example"; }
```
  - Visual: A sticky note attached to code, providing extra instructions or notes.

22. **Lambda Expressions**:
  - Example:
```java
(int a, int b) -> a + b;
```
  - Visual: A shorthand way of writing a function, like a formula.

23. **Functional Interface**:
  - Example:
```java
@FunctionalInterface
interface MyFunction {
   int apply(int x);
}
```
  - Visual: A single-page document that describes one specific task (method).

24. **Date & Time Packages**:
  - Example:
```java
LocalDate today = LocalDate.now();
System.out.println(today);
```

   - Visual: A modern calendar that gives you precise date and time data.

25. **Memory Management - Heap vs Stack**:
   - Example:
     - Stack: Local variables and method calls are stored here.
     - Heap: Objects and instance variables are stored here.
   - Visual: Think of the stack as a small desk for quick tasks (limited space), and the heap as a larger storage room (more space, but slower to access).

26. **Best Practices**:
   - Example:
     - Follow naming conventions, write clean and readable code, use version control.
   - Visual: A checklist for ensuring your code is organized, efficient, and maintainable.

If you need further elaboration on any of these examples, or if you'd like to discuss specific code implementations, feel free to ask!