BÀI TẬP PHẦN 1

BÀI 1. Sử dụng Set trong collection để tìm tập giao và tập hợp giữa 2 tập hợp (*tự tạo ra 2 tập hợp, mỗi tập hợp 2.000.000 một số Integer ngẫu nhiên, không giống nhau. 2 tập hợp phải có trên 5% số phần tử giao nhau*). Hãy tính tổng thời gian thực hiện tìm cả tập giao và tập hợp và tìm cách tối ưu nhất để thời gian tính toán này nhỏ nhất có thể.

BÀI 2. Cho một văn bản như trong file "input 2.zip". Hãy đọc file và đếm số lần xuất hiện của từng từ (Sử dụng stringtokenizer để tách từ, Các từ này được tách với nhau bằng các dấu sau " .,!=+-") rồi ghi vào file output.txt.

BÀI 3. Lập trình đếm từ đa luồng cho bài 2 với dữ liệu vào là một folder chứa nhiều file text được nén trong file "input 3.zip" bằng cách sử dụng Callable và Future. Hãy xử lý song song các file và tìm top 10 từ xuất hiện nhiều nhất, và top 10 từ xuất hiện ít nhất của toàn bộ dữ liệu có trong folder. Lưu ý, chỉ được chạy tối đa 6 luồng cùng lúc.

BÀI 4. Tạo ra một class Point với 2 tọa độ nguyên x và y. Sinh ngẫu nhiên 30.000 point khác nhau, nằm trong 3 Set. Set thứ nhất có size 8.000, chứa các điểm có cách điểm A(800, 800) một độ dài không quá 400 đơn vị, Set tiếp theo có size = 10.000 chứa các điểm cách điểm B(4000,800) không quá 500 đơn vị, và 12.000 điểm cuối cùng cách điểm C(2400, 2400) không quá 600 đơn vị. Trộn ngẫu nhiên 30.000 điểm này, sau đó ghi ra file output4.txt. Lưu ý: sử dụng hàm Set< Point >.contains để kiểm tra một điểm đã tồn tại trong tập hợp đó hay chưa (kiểm tra kỹ lại check 2 điểm có tọa độ giống nhau).

BÀI 5. Trong file Maze.java có chứa bản đồ mê cung, hãy implement hàm `solve` dựa trên một phương pháp tìm đường nào đó để tìm đường đi đến điểm đánh dấu ở giữa bản đồ. Hãy chọn các phương pháp có độ phức tạp (số bước thực hiện) càng nhỏ càng tốt và đánh dấu đường đi bằng cách thay số 0 bằng số 2 trên ma trận mê cung. Lưu ý, bài tập này có xét cách viết code, comment, viết java doc, đặt tên biến/hàm, tổ chức project …

BÀI 6. Thiết kế hướng đối tượng
Tạo các đối tượng (class) cho một hệ thống xuất hóa đơn của một cửa hàng tạp hóa. Hệ thống bao gồm các đối tượng sau:
Khách hàng (mã khách hàng, giới tính, độ tuổi)
Nhân viên bán hàng (mã nv, giới tính, ngày làm việc, ca đăng ký)
Nhân viên nhập hàng (mã nv, giới tính, ngày làm việc, thâm niên)
Mặt hàng (mã hàng hóa, tên hàng hóa, phân loại, giá)
Hóa đơn (mã hóa đơn, mã nv bán hàng, mã KH nếu có, danh sách mặt hàng, tổng giá, ngày mua)

BÀI 7. Một url là url thỏa mãn các yếu tố (ví dụ 1 url hợp lệ: https://tiki.vn/dien-thoai-may-tinh-bang/c1789?src=mega-menu):
- Bắt đầu bằng http hoặc https
- Có thể chứa www hoặc không
- Tên miền của url (ví dụ tiki) chỉ chứa các ký tự la tinh hoa/thường từ a-z và chữ số.
- Phần domain extension (ví dụ .vn) phải chứa dấu chấm ở đầu, thêm sau là các ký tự in thường có độ dài từ 2 đến 6 ký tự.
- Phần path (ví dụ /dien-thoai-may-tinh-bang/c1789?src=mega-menu) không được chứa dấu cách.

Viết đoạn regex để kiểm tra xem một url bất kỳ có là hợp lệ hay không.

BÀI 8. Cho hai lớp Country và City có các thuộc tính (như bên dưới). Một quốc gia có 1 thủ đô và có thể có 1 hoặc nhiều thành phố. Một thành phố luôn trực thuộc chỉ 1 quốc gia.
- Country:
  + String code (mã quốc gia)
  + String name
  + String continent (mã lục địa)
  + double surfaceArea (diện tích bề mặt)
  + int population (dân số)
  + double gnp (Gross National Product)
  + int capital (mã thành phố là thủ đô của đất nước này)

- City:
  + int id
  + String name
  + int population (dân số của thành phố)

- Dữ liệu về các quốc gia, thành phố được lưu trong 2 file countries.dat và cities. Hãy thực hiện đọc thông tin các thành quốc, quốc gia từ file
sau đó sử dụng lambda và stream api trong java 8 để thực hiện các yêu cầu sau:

1.1 Tìm thành phố đông dân nhất của mỗi quốc gia.
1.2 Tìm thành phố đông dân nhất của mỗi lục địa.
1.3 Tìm thành phố là thủ đô, đông dân nhất.
1.4 Tìm thành phố là thủ đô, đông dân nhất của mỗi lục địa.
1.5 Sắp xếp các quốc gia theo số lượng thành phố giảm dần.
1.6 Sắp xếp các quốc gia theo mật độ dân số theo thứ tự giảm dần bỏ qua các quốc gia có dân số bằng không.