El present exercici gira al voltant del mon de la **fórmula 1**, ja que, independentment de si és esport o no, si ens agrada o no, és un gran generador de dades. Aquestes dades es convertiran en estadístiques i molts programes les s'han de analitzar.

Hem rebut dos arxius de la temporada 2017, i una sèrie de peticions a fer sobre aquests arxius. De la nostra habilitat dependrà o no que ens contractem a un famós equip.

Arxiu ***circuits.json***: conté informació de tots els circuits on s'han corregut grans premis. El format és el que es veu a continuació:

```
{
    "circuitId": "sepang",
    "url": "http://en.wikipedia.org/wiki/Sepang_International_Circuit",
    "circuitName": "Sepang International Circuit",
    "Location": {
        "lat": "2.76083",
        "long": "101.738",
        "locality": "Kuala Lumpur",
        "country": "Malaysia"
    }
},
```

Arxiu **monaco_2017.xml**, que conté la informació de la cursa. La part que ens interessa és els elements Result, que són la informació de cada participant de la cursa:

```
<ResultsList>
    <Result number="5" position="1" positionText="1" points="25">
        <Driver driverId="vettel" code="VET" url="http://en.wikipedia.org/wiki/Sebastian_Vettel">
            <PermanentNumber>5</PermanentNumber>
            <GivenName>Sebastian</GivenName>
            <FamilyName>Vettel</FamilyName>
            <DateOfBirth>1987-07-03</DateOfBirth>
            <Nationality>German</Nationality>
        </Driver>
        <Constructor constructorId="ferrari" url="http://en.wikipedia.org/wiki/Scuderia_Ferrari">
            <Name>Ferrari</Name>
            <Nationality>Italian</Nationality>
        </Constructor>
        <Grid>2</Grid>
        <Laps>78</Laps>
        <Status statusId="1">Finished</Status>
        <Time millis="6284340">1:44:44.340</Time>
        <FastestLap rank="2" lap="38">
            <Time>1:15.238</Time>
            <AverageSpeed units="kph">159.669</AverageSpeed>
        </FastestLap>
    </Result>
    <Result number="7" position="2" positionText="2" points="18">
        <Driver driverId="raikkonen" code="RAI" url="http://en.wikipedia.org/wiki/Kimi_R%C3%A4ikk%C3%
            <PermanentNumber>7</PermanentNumber>
            <GivenName>Kimi</GivenName>
            <FamilyName>Räikkönen</FamilyName>
```

VALORACIÓ GLOBAL: Es valorarà amb **1 punt** el fet de programar de manera neta:

1. Encapsulació i creació adecauda de variables
2. Comentaris
3. Tractament d'excepcions
4. Evitar errors de compilació

EXERCICI 1. Els Circuits (4 punts)

Hem de carregar la informació del circuits per a processar-la. Com hauràs observat, dins de cada circuit existeix una localització (Location), per la qual cosa necessitarem les següents classes:

- **Location** → aquesta classe es passa parcialment implementada. Els atributs, getters, setters, i toString ja estan fets. **Has d'implementar un constructor que rep un objecte JSON de la localització per a crear-la**. Ademès s'incorpora un mètode distanciaTo(Location altre) ja implementat que calcula la distància entre dos Location.
- **Circuit** → Has d'implemetar **tota la classe amb el que es dona**. Contindrà els següents atributs:
  - private String name;
  - private Location loc;
  - private String url;

Ha de contenir també un constructor a partir d'un JSONObject. Pots ajudar-te de les generacions de codi que fan els IDE

Es demana:

1. Implementar **Constructor de la classe Location** a partir d'un objecte JSON.

2. Implementar la classe **Circuit** seguint els requeriments que es demanen.

3. Mètode **ArrayList<Circuit> carregaCircuitsJSON(String nomJSON){}** que li passarem el nom del fitxer (***circuits.json***) i ens retorna un Array de Circuit amb tots els circuits del fitxer

4. Mètode **void imprimirCercanos(ArrayList<Circuit> elsCircuits, Location l, double km){}**, que rep l'array anterior, una Localització i una quantitat. Mostrarà aquells circuits que la seua distància a la localització passada sigui menor que la quantitat del tercer argument. Per a provar-ho, al projecte s'ha creat la següent variable:

   ```
   public static final Location TAVERNES =
      new Location(39.08357, -0.24115, "Tavernes de la Valldigna", "Espanya");
   ```

   Es demana provar-ho mostrant els circuits que disten de TAVERNES menys que 1000 km.

5. Mètode **guardarObjectes(ArrayList<Circuit> elsCircuits, String nomObjectes) {}**, que guarda en un fitxer d'objectes tots els circuits.

EXERCICI 2. La cursa (5 punts)

Al fitxer **monaco_2017.xml** es presenta el resultat de la cursa. Entre altres coses, aquest fitxer conté una col·lecció de **Result**, que representa aquells que han participat en la carrera. Dins del Result trobarem com atributs el número del pilot (***number***) i la posició en la que ha acabat la carrera (***position***). Ademès trobarem els següents elements:

- **Driver**: informació de quí és el conductor
- **Constructor**: informació de la marca del cotxe que condueix
- **Grid**: Posició en la que ha sortit el conductor
- **Laps**: Voltes que ha completat
- **Status**: que ens indica amb l'atribut ***statusID***=1 que ha acabat la carrera
- **Time**: que ens indica mb l'atribut ***millis*** el que ha tardat en completar la carrera (en ms), i en el seu valor la diferència respecte als primer.
- **FastesLap**: que ens diu la classificació respecte a la volta ràpida al seu atribut ***rank***.

Nota: La classe **Driver** està totalment implementada. Sols necessitarà passar-se un element XML amb la informació del conductor (el quadre roig de la primera fulla) al constructor per a crear un objecte Driver.

La classe **ResultadoCarrera** que es passa parcialment implementada conté aquest atributs:

```
private Driver d;
private String constructor;
private int initialPos;
private int finalPos;
private long timeMillis;
private int completedLaps;
private int rankFastesLap;
private boolean finisher;
```

- Es demana implementar a la classe **ResultadoCarrera:**

  1. mètode **Constructor**, que rebrà un element XML de tipus **Result** (quadre verd de la primera fulla) per a omplir tots els camps demanats. CONSELL: De no saber fer-ne algun, deixa'l i inicia a algun valor per defecte, millor que no fer res.

  2. mètode **public String toCSV(){}**, que retornarà una representació del resultat amb tots el camps separàts pel caràcter punt i coma (**;**)

- Es demana implementar al programa principal:

  3. Mètode **ArrayList<ResultadoCarrera> carregaResultadosXML(String nomXML){}** que rep el nom del fitxer XML i recuperarà tots els resultats que hi han.

  4. Mètode **void saveAsCSV(String nomFitxer, ArrayList<ResultadoCarrera> elsResultats){}** que guarda els resultats prèviament carregats a un fitxer CSV.

NOTA:

- POTS afegir les funcions que consideres
- NO POTS modificar els prototips de les funcions que es demana implementar
- El main que et passa el professor NO L'HAS DE MODIFICAR, i cal executar-lo sense donar errors.
- Pots consultar tota la documentació de classe i d'internet
- No poden fer-se servir programes de missatgeria. Si es detecta alguna mena d'irregularitat implicarà un zero a l'examen.
- Entregar un fitxer **zip** amb la carpeta de projecte