A projekt méretére tekintettel nem kerül minden file átvizsgálásra.

Conroller.java

* A kód Javadoc használatával van kommentezve, ami segíti az olvashatóságot és segíti a kód megértését.
* A kommentek magyarul vannak írva, ami nehezítheti az együttműködést esetleges új fejlesztők bevonásánál.
* A változók és metódusok elnevezése nem követ konzisztensen egy konvenciót, ami félreértésekhez vezethet.  
  Néhány példa:  
  CoolDown – roundsLeft – plumberCoolDown - setRounds\_min1  
  Javítás:  
  A feklesztő csapatnek meg kell egyeznie egy egységes konvencuóban, amit hasznláni fognak.   
  Javaslat:  
  camelCase, másik ismert nevén PascalCase használata
* Néhány metódus elnevezése nem illeszkedik a programozásban megszokott nevekhez, így félreértésekhez is vezethetnek:  
  Példa: setRounds\_min1()  
  Javítási javaslat: decrementRounds()
* Az osztály egy minden változója és metódusa statikus, feltehetőleg singleton- nak szánták, de ez nem került implementálásra, de sem megerősítésre, vagy magyarázatra nem került a kommentekben.  
  Probléma:  
  Az osztályt használó fejlesztők nem ismerik feltétlenül az osztály implementációját, így akaratunk ellenére többször is példányosításra kerülhet az osztály, ami nemvárt viselkedéshez vezethet.  
  Javítás/Javaslat:  
  A singelton terezési minta megvalósítása. Egy statikus tagváltozó létrehozása, ami eltárolja az egy példányt, és a publikus konstruktor priváttá tevése és egy publikus függvénnyel való elfedése, ami az említett tagváltozóban tárolt példányt adja vissza, amennyiben nincs még ilyen akkor meghívja a konstruktort. Ezzel megelőzhetjük, hogy az osztály többször is példányosításra kerüljön.
* A setterek nem ellenőrzik a megadott értékek helyességét, így akár az értelmezési tartományon kívül eső értékek is beállításra kerülhetnek.  
  Példa:

public static void setRounds(int i) {

roundsLeft = i;

}

Ennél a példánál például nem ellenőrizzük az érték helyességét, így negatív értékek is beállításra kerülhetnek, ami nem várt viselkedéshez vezethet.

Javítás:   
public static void setRounds(int i) {

if (i < 0) {

throw new IllegalArgumentException("Round count cannot be negative");

}

roundsLeft = i;

}

* Az osztály megvalósítása nem thread safe, ami az alkalmazás futásakor hibákhoz vezethet, amennyiben fent áll a lehetősége, hogy több szál is el akarja érni az osztályt célszerű a metódusok aszinkronná tétele.
* A turn() metódusban rengetek felesleges kikommentezett kód található, ami rontja a kód olvashatóságát.
* A turn() metódusban a logika jelentős része duplán van jelen, külön lett implementálva a Nomad és a Plumber osztályokra, ez összevonható.

Field.java

* Feleslegesen importált könyvtárak eltávolítása.  
  Példa: import java.awt.image.BufferedImage; nem kerül használatra
* A kód javadoc használatával került kommentezésre, ami segíti az olvashatóságot, de a kommentezés helyenként befejezetlen, hiányos, ami nehezebben értelmezhetővé teszi.
* A kód sok felesleges kikommentezett kódot tartalmaz, ami rontja az olvashatóságát.   
  Javítás:  
  Ezen sorok egyszerű törlése.
* A változók és metódusok elnevezése nem követ konzisztensen egy konvenciót, ami félreértésekhez vezethet.  
  Néhány példa:  
  SetName() – toString() – destroy() – SpareWater()  
  Javítás:  
  A feklesztő csapatnek meg kell egyeznie egy egységes konvencuóban, amit hasznláni fognak.   
  Javaslat:  
  camelCase, másik ismert nevén PascalCase használata
* Az ArrayListet visszaadó metódusok nem mindegyike ad vissza generikusan specifikált ArrayListet így például a getConnectedPipe() és getConnectedPumps() metódusok sem.  
  Javítás:  
  public abstract ArrayList<Pipe> getConnectedPipes();  
  public abstract ArrayList<Pump> getConnectedPumps();
* Hiányos metódus deklarációk, így pédául a feltehetőleg absztraktnak szánt waterFlowing() metódus nem absztrakt.