Potrebno je napisati klasu `FactorialIterator` koja implementirata sučelje `java.util.Iterator<Integer>`. Klasi se u konstruktoru prosljeđuje koliko brojeva će generirati. Ako se pošalje negativan broj potrebno je baciti iznimku.

Klasa ima vidljivost postavljenu na *package private*.

Primjer korištenja:

```
Iterator<Integer> iterator = new FactorialIterator(10);
while(iterator.hasNext())
    System.out.println(iterator.next());
```

Očekivani ispis:

```
1
1
2
6
24
120
720
5040
40320
362880
```

Neka su zadane apstraktne klase:

```
abstract class Game {
    private String name;
    private int rating;

    protected Game(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public int getRating() {
        return rating;
    }

    public void setRating(int rating) {
        this.rating = rating;
    }

    @Override
    public String toString() {
        return name + ":" + rating;
    }
}

abstract class Player {
    private String name;

    protected Player(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }
```

```
    abstract void addGameScore(Game game, int score);

    abstract Iterable<Integer> getScores(Game game);
}
```

Napisati sve što je potrebno da bi:

a) se uspješno izveo sljedeći kod

```
Game g1 = Factory.createGame("Super Mario", 90);
Game g2 = Factory.createGame("Heroes", 80);
MyPlayer p = new MyPlayer("Mario");
p.addGameScore(g1, 300);
p.addGameScore(g1, 400);
p.addGameScore(g2, 50);
p.addGameScore(g1, 200);
p.addGameScore(g1, 400);
p.addGameScore(g2, 70);
for(Game g : p) {
    for(int score : p.getScores(g)) {
        //do something
    }
}
```

b) iteriranje po igrama išlo po nazivu igre (prvo Heroes s 50 i 70, a onda Super Mario s 300, 400, 200 i 400), a iteriranje po ostvarenim rezultatima u igri po redoslijedu kojim je rezultat ostvaren

Predati kod klase `Factory` i svih ostalih klasa potrebnih za realizaciju zadatka (sve klase predati bez modifikatora vidljivosti). Klase `Game` i `Player` ne mogu se mijenjati i njih ne predajete.

Napomena: Sve klase i sučelja imaju vidljivost postavljenu na package private.

Napomena: Prije predaje iz Vašeg koda izbacite sve ispise na standardni izlaz kako bi testovi ispravno radili.

U ovom zadatku se koristi zapis decimalnih brojeva na "distribuirani" način. Njihove znamenke su zapisane u listi sa oznakama pozicije. Detaljnije:

- Lista se sastoji od **podlista** koje sadrže po dva elementa (**a**,**b**).
- pritom **a** označava redni broj decimalnog mjesta (pozicije) na sljedeći način: 0.1234567. 0. je zadnje mjesto ispred decimalne točke, a potom ostali redni brojevi označavaju pozicije na desno od decimalne točke. Pozicije lijevo od 0. se ignoriraju.
- pritom **b** je znamenka na mjestu označenom sa pripadnim **a**.

Primjer: broj 3.141 zapisan na distribuirani način jest: ((0,3), (1,1), (2,4), (3,1))

Nadopunite odsječak koda:

```
class Solution{

    public static Predicate<List<List<Integer>>> allDigitsMatch(double exemplar){
        return  /* ovdje dodati rjesenje */
    }


    public static Predicate<List<List<Integer>>> allDigitsDefined(){
        return   /* ovdje dodati rjesenje */
    }

}
```

prema sljedećim zahtjevima:

- metoda `allDigitsMatch` vraća predikat koji omogućuje testiranje odgovara li zapis znamenki u listi znamenkama priloženog decimalnog broja.
- metoda `allDigitsDefined` vraća predikat koji omogućuje testiranje jesu li zapisu znamenki u listi zapisane sve uzastopne znamenke.
- čitavu klasu `Solution` zalijepiti u Edgar.

Primjer

```
List<List<Integer>> ulaz1= Arrays.asList(Arrays.asList(1,2),Arrays.asList(0,3),Arrays.asList(2,6))); // 3.26

boolean t1 = Solution.allDigitsMatch(3.266).test(ulaz1); // true
boolean t2 = Solution.allDigitsDefined().test(ulaz1); // true

List<List<Integer>> ulaz2=
Arrays.asList(Arrays.asList(1,2),Arrays.asList(0,3),Arrays.asList(4,6))); // 3.2**6
boolean t1 = Solution.allDigitsMatch(3.266).test(ulaz2); // true
boolean t2 = Solution.allDigitsDefined().test(ulaz2); // false
```

NAPOMENA: pretpostavite da su testovi takvi da su svi redni brojevi nenegativni i jedinstveni, te znamenke jednoznamenkaste.