Data Set: 10,000 Movie Lens ratings from about 1,000 users, 2000 movies.
Rating Scale: 1, 2, 3, 4, or 5.

Toy user-item utility matrix

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

Recommender was built using a Pearson correlation coefficient as a similarity metric between user rating vectors.

$$c_{a,u} = \frac{\text{covar}(r_a, r_u)}{\sigma_{r_a}\sigma_{r_u}}$$

r_a and r_u are the ratings vectors for the *m* items rated by **both** *a* and *u*

In practice a uniform distribution is assumed for ratings: Given a random movie a person is just as likely to rate that movie a 1, 2, 3, 4, or 5. This assumption simplifies the equation. Now we can express it in a form that shows the significance of this metric in the present context:

Let $\mu_x =$ the average of ratings in $\mathbf{r_x}$.

$$\text{Let } \mathbf{r'_x} = \mathbf{r_x} - \begin{bmatrix} \mu_x \\ \vdots \\ \mu_x \end{bmatrix}.$$

$$\text{Then, } c_{a,u} = \frac{\mathbf{r'_a} \cdot \mathbf{r'_u}}{\|\mathbf{r'_a}\|\|\mathbf{r'_u}\|}.$$

Calculation of Rating prediction

$$p_{a,i} = \bar{r}_a + \frac{\sum_{u=1}^{n} w_{a,u}(r_{u,i} - \bar{r}_u)}{\sum_{u=1}^{n} |w_{a,u}|}$$

Which is just the cosign of the angle between our “normalized” ratings vectors (obtained by subtracting the mean rating of the vector from each rating in the vector). This gives a similarity score between -1 and 1, where scores close to zero signify little correlation, scores closer to 1 signify that the two users rate items more similarly, and scores closer to -1 signify that the two users rate items more differently. A weight may be employed when constructing a metric to discount vectors with few entries.

The Recommender

```java
DataModel model = new FileDataModel(new File("data/movies.csv"));
PearsonCorrelationSimilarity similarity = new PearsonCorrelationSimilarity(model);
UserNeighborhood neighborhood = new ThresholdUserNeighborhood(0.1, similarity, model);//(threshold, si
UserBasedRecommender recommender = new GenericUserBasedRecommender(model, neighborhood, similarity);

int x = 1;
for(LongPrimitiveIterator users = model.getUserIDs(); users.hasNext();) {
        long userId = users.nextLong();
        List<RecommendedItem>  recommendations = recommender.recommend(userId, 5);
        System.out.println(userId + ": ");
    for (RecommendedItem recommendation : recommendations) {
        System.out.println(recommendation.getItemID()  + "\t" + recommendation.getValue() + "\n");
    }
    x++;
    if (x>10) System.exit(1);
}
```

Sample Output

```
1:
1558  5.0
1500  5.0
1467  5.0
1189  5.0
1293  5.0

2:
1643  5.0
1467  5.0
1500  5.0
1293  5.0
1189  5.0

3:
1189  5.0
1500  5.0
1302  5.0
1368  5.0
1398  4.759591

4:
1104  4.7937207
853   4.729132
169   4.655577
1449  4.60582
408   4.582672

5:
1500  5.0
1233  5.0
851   5.0
1189  5.0
119   5.0

6:
1467  5.0
1189  5.0
1293  5.0
1398  4.8224106
1592  4.7151284

7:
1500  5.0
1467  5.0
1293  5.0
1189  5.0
1125  4.734744

8:
1467  5.0
1293  5.0
1189  5.0
1612  4.582285
169   4.5788593
```