# User Study Task (Make a Copy First!)

You have 15 minutes to complete this task. The task is not to test how quickly you can finish the task. So please aim for the thoroughness, not speed. The experimenter will remind you when you have 5 minutes left.

**[5 min] Part I. Get familiar with the Stack Overflow post and its code example.**

Suppose you want to implement a function that computes the distance between two coordinates on earth. You find [this SO post](https://stackoverflow.com/questions/837872/calculate-distance-in-meters-when-you-know-longitude-and-latitude-in-java/837957#837957) that illustrates how to implement the function in Java. Please inspect the code example in this post and focus on understanding its functionality and implementation details.

*Experimenter: Are you confident about your understanding of this function to move to the next step?*

**[10 min] Part II. Create a reusable template of the SO code example.**

Now you want to create a template of the SO code example so other programmers do not have to repetitively adapt the code example to their own projects.

Please specify the following code parts in the given code example.

(1) Use the blue color to tag where you consider as the gist of the example and should remain unchanged during code reuse.

(2) Use the yellow color to tag where should be changed or removed based on the context of the target program.

(3) Use the red color to specify what additional code you would like to add into the template.

|  |
| --- |
| Group lat1 and lng1 GeoPoint  public static float distFrom(float lat1, float lng1, float lat2, float lng2) {  double earthRadius = 6371000; //meters P15: Move down to usage  double dLat = Math.toRadians(lat2-lat1);  double dLng = Math.toRadians(lng2-lng1); P15: Do computation on GeoPoint  double a = Math.sin(dLat/2) \* Math.sin(dLat/2) P15: repeated comutation  + Math.cos(Math.toRadians(lat1)) \* Math.cos(Math.toRadians(lat2))  Do computation on geopoint  \* Math.sin(dLng/2) \* Math.sin(dLng/2); P15: repeated computation  double c = 2 \* Math.atan2(Math.sqrt(a), Math.sqrt(1-a));  float dist = (float) (earthRadius \* c);  return dist;  } |

1. Rename a to a more readable and meaningful name (refactoring)
2. Rename c to a more readable and meaningful name (refactoring)
3. Move the declaration of earthRadius closer to its usage (refactoring)
4. Refactor float lat1, float lng1 and float lat2, float lng2 to GeoPoint (refactoring)
5. Refactor Math.sin(dLat/2) \* Math.sin(dLat/2) to avoid repeated computation
6. Refactor Math.cos(Math.toRadians(lat1)) \* Math.cos(Math.toRadians(lat2)) to avoid repeated computation
7. Refactor Math.sin(dLng/2) \* Math.sin(dLng/2) to avoid repeated computation