Explain your understanding of methods, objects, classes, and the object-orientated nature of Java with the help of segments of codes. Avoid using examples given in the course materials. You can write your own codes or comment on codes written by your classmates.

Methods are a step by step procedure to accomplish a task. One point I argue in explaining the difference between a method and a function is that a method can have alternating input(s) and an output while functions have an expected output based on expected input(s). Here’s an example of a class method that takes in a Scanner object as an argument in its paramaters does some work and returns Map data structure that contains the key, value pairs of String, Integer accordingly:  
  
**public** Map<String, Integer> getWords(Scanner input)

{

Map<String, Integer> wordsCount = **new** TreeMap<String, Integer>();

input.useDelimiter("[^a-zA-Z]+");

**while** (input.hasNext() )

{

String next = input.next().toLowerCase();

**if** (wordsCount.containsKey(next)) {

**int** count = wordsCount.get(next);

wordsCount.put(next, count + 1);

} **else** {

wordsCount.put(next, 1);

}

}

**return** wordsCount;

}

Objects are instances of classes. They use the ‘new’ keyword in Java and are allocated on the Heap memory. Here’s an example of a declaration and instantiation of an Object:

CommonWords cw = **new** CommonWords();

cw.determineCommonWords(**new** File("C:/Users/Odiscious/Downloads/Hamlet.txt")

, **new** File("C:/Users/Odiscious/Downloads/KingLear.txt"));

Classes are what I consider as the blueprint of an object. With the use of private data members it is possible to encapsulate data in a class object. Taking a closer look at our Book class from a previous assignment, we can see that we built accessor and mutator methods to help encapsulate data in the Book object. By doing this, we have limited what the caller of the Book object can modify and more importantly, how they will modify it. In scaling the solution, we could modify the Book class to not have mutator functions and thereby limit the caller to setting the field variables only once per object. Here’s an example:  
  
public class Book {

private String title;

private String author;

private Person person;

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public String getAuthor() {

return author;

}

public void setAuthor(String author) {

this.author = author;

}

public Book(String nameOfBook) {

// TODO Auto-generated constructor stub

author = "unknown author";

title = nameOfBook;

}

public Person getPerson() {

return this.person;

}

public void setPerson(Person p2) {

this.person = p2;

}

public String toString() {

String available;

if(this.getPerson() == null)

{

available = "Available";

}

else{

available = "Checked out to " +

this.getPerson().getName();

}

return this.getTitle() + " by " + this.getAuthor() + "; " + available;

}

}