1. What does it mean to overload a method? Why is it useful?

To give the same name to two or more methods in the same class or in classes related by inheritance. Overloaded methods must either have different numbers of parameters or have some parameter position that is of differing types in the method headings (i.e., the methods must have different signatures). It is useful because it enables related methods to be called by using a common name. For example, you can have a method that returns the absolute value of an `int`, and another one that returns the absolute value of a `double`. You can name both of these methods based on their function: `abs` (in fact, there are several `abs` methods in Java’s `Math` class). Based on the type of the argument you pass to the method, the compiler can decide which method to use.

2. Write two `findIndex` methods such that the method name is overloaded. One method should take a 1D array of integers as a parameter, and an `int x` and return the index of `x` in the array, and the other method should print a 1D array of Strings and a `String s`, and return the index of `s` in the array. Both methods should return `-1` if the value is not in the array.

```java
public int findIndex(int[] a, int x) {
    for (int i = 0; i < a.length; i++) {
        if(a[i] == x)
            return i;
    }
    return -1;
}

public int findIndex(String[] a, String s) {
    for (int i = 0; i < a.length; i++) {
        if(a[i].equals(s))
            return i;
    }
    return -1;
}
```

3. What does it mean to override a method? Why is it useful?

In a derived class, if you include a method definition that has the same name, the same number, types, and order of parameters, and the same return type as a method already in the base class, this new method definition overrides the method defined in the base class. This new definition replaces the old definition of the method when objects of the derived class receive a call to the method. This enables you to put common functionality in base classes, and more specialized functionality in derived classes, but still use the same method name and parameters to perform the specialized functionality.

4. What is wrong with this code?
int[][] table = new int[15][7];

for (int row = 0; row <= table.length; row++)
    for (int col = 0; col <= table[row].length; col++)
        table[row][col] = 0;

You will get an ArrayIndexOutOfBoundsException. Array indices are 0-based. Use < table.length and < table[row].length, not <=.

5. Write a method that takes an array of doubles as a parameter and returns the average of the elements.

public double average(double[] array) {
    if (array.length == 0) {
        return 0.0;
    }

double sum = 0.0;
for (int i = 0; i < array.length; i++) {
    sum += array[i];
}
return sum / array.length;
}

6. Assume you have the following class, with only one method and one constructor defined:

public class Student {
    private String name;
    public Student(String name) {
        this.name = name;
    }
    public String getName() {
        return name;
    }
}

Would the following code compile (in particular, the line that calls the equals method)? Why or why not?

Student std = new Student("Apu");
Student std2 = new Student("James");
if (std.equals(std2))
    System.out.println("They are equal");

Yes, it would compile. Every class in Java is a subclass of the class Object. The Object class defines an equals method, so the Student class inherits it. Therefore, the code
The code may not, however, do what you intend it to do (assuming you intend for two Students to be equal if they have the same name). You would have to override Object’s equals method with your own equals method inside of the Student class for it to work correctly.

7. Given the following classes, draw a diagram showing which classes would inherit methods from other classes.

Dog   Quadruped  Creature  Elephant  Biped  Human  Kangaroo

```
Creature

  Quadruped
  |
  Biped

  Dog
  |
  Elephant
  |
  Human
  |
  Kangaroo
```

8. Given your diagram above, are the following lines of code legal or illegal?

- _legal_ Quadruped q = new Quadruped();
- _illegal_ Quadruped q2 = new Biped();
- _legal_ Biped b = new Human();
- _illegal_ Elephant e = new Creature();
- _legal_ Creature c = new Kangaroo();
- _legal_ Dog d = new Dog();

Think about the is-a relationship when determining if the above statements are legal.

9. Which class from #7 would contain these methods?
What about hop()?

It depends
Hopping would probably be most relevant for a kangaroo, but other creatures can hop too. When you design different classes, it’s important to think about what classes need certain functionality. If, in your design, it does not make sense for anything but a kangaroo to hop, then the method should probably go there. If all bipeds can hop, then it should go there, and so on.

10. What is the output of this code?

```java
public static void doSomething(int[] arr) {
    int[] myArr = arr;
    myArr[0] = 37;
}
public static void main(String[] args) {
    int[] arr = { 3, 6, 10, 17 };
    doSomething(arr);
    for (int val : arr) {
        System.out.println(val / 2);
    }
}
```

18
3
5
8

After the line `int[] myArr = arr;` in the `doSomething` method, the variables `arr` and `myArr` both contain a reference to the same array in memory. Therefore, changing `myArr`’s elements also changes the elements referenced by `arr`.

11. Consider the following code.
public static void changeArray(int[][] arr)
{
    for (int row = 0; row < arr.length; row++)
        for (int col = row; col < arr[row].length; col++)
            arr[row][col] = row + col;
}

What is contained in the 2D array myArray after the following code?
int[][] myArray = new int[5][5];
changeArray(myArray);

0 1 2 3 4
0 2 3 4 5
0 0 4 5 6
0 0 0 6 7
0 0 0 0 8

Note that the inner loop’s initialization statement sets the value of col to the value of row, so the lower left triangular area of the 2D array does not change.