Q2 [2 points]

```
class Person {
  String name;
  Person(String s) { name = s; }
}

interface TeachingAssistant {
  public void grade();
}
```

Considering the code above, write a minimal class `Student` which is derived from `Person` and implements `TeachingAssistant`.

Q6 [1 point]

Explain some of the differences between a standard *class* and an *interface*.

Q7 [1 point]

What does it mean that a method `f()` may *throw an exception*, and how would you call `f()` in that case?

Q9 [2 points]

What is the difference between the `Comparable` and `Comparator` interfaces? When might you use one instead of the other?

Q10 [2 points]

Suppose you have a class `Shape` which contains the abstract method `computeVolume()`, and you derive the concrete classes `Sphere` and `Cube` from it. Let `Sphere` add an instance variable `radius` and `Cube` add a variable `sideLength` which are given default values in their constructors.

(a) What is the name of the language feature that makes this line of code legal?

```
Shape s = new Sphere();
```

(b) If this were the next line after the code in **(a)**, would it compile?

```
s.computeVolume();
```

(c) Would this compile?

```
Cube c = new Cube();
c.computeVolume();
```

(d) If this were the next line after the code in **(a)**, it would not compile. How could you fix it with *casting*?

```
System.out.println(s.radius);
```