Bryan Zhang
1. Put the following in chronological order for a simple program:
Runtime - third
Programming (Writing the code) - first
Compilation - second

2. Put the following in order in a method header:
Parameters
Return Type
Method Name
Modifiers
Modifiers - return type - method name - parameters
3. Primitive data types store data, while references store the address of an object.

4. What are the toString and equals methods? Does every class have them?
They are default methods inherited from Object, every class has them.
5. Can a class inherit from multiple superclasses? Can a class implement multiple interfaces? Do classes need to have extends Object in their header?
A class can only inherit from one superclass. A class can implement multiple interfaces. Classes do not need to have extends Object in their header.
6. If the Employee class extends the Person class, then an Employee is a Person. Does this relation hold in the other direction?
No.
7. Can abstract classes be instantiated? If not, what is their use?
Abstract classes cannot be instantiated. It is mostly used to create a base which can be extended by multiple subclasses which can override the abstract methods.
8. Can inner classes access private variables in their outer classes?
Yes.
9. Methods in an interface are implicitly public and abstract.

10. What are the benefits of using generic classes?
You can use types as parameters. This eliminates any casting when retrieving objects.
11. Can you use a type parameter's constructor? Can you create an array of a generic type?
No to both.
12. Create a class named Point3D that implements Comparable<Point3D>. The natural order is based on first the x-coordinate, then the y-coordinate if tied, then the z-coordinate if tied.

```
import java.util.*;
public class Point3D implements Comparable<Point3D>
{
	private int x;
	private int y;
	private int z;
	public Point3D(int x, int y, int z)
```

```
	{
		this.x = x;
		this.y = y;
		this.z = z;
	}
	public int compareTo(Point3D p)
	{
		if (x > p.x) {return 1;}
		if (x < p.x) {return -1;}
		if (y > p.y) {return 1;}
		if (y < p.y) {return -1;}
		if (z > p.z) {return 1;}
		if (z < p.z) {return -1;}
		return 0;
	}
}
```

13. Create an array of type Point3D with at least 5 elements and sort it.

```
import java.util.Arrays;
public class Point3DTest
{
	public static void main(String[] args)
	{
		Point3D a = new Point3D(0,4,2);
		Point3D b = new Point3D(5,1,-6);
		Point3D c = new Point3D(3,-1,5);
		Point3D d = new Point3D(-2,4,6);
		Point3D e = new Point3D(5,10,2);
		Point3D[] array = {a,b,c,d,e};
		Arrays.sort(array);
	}
}
```

14. Create a method that takes an array of an arbitrary Comparable type and returns the minimum element in the array (minimum element is the first element when sorted).

```
	public static Point3D minElement(Point3D[] array)
	{
		Arrays.sort(array);
		return array[0];
	}
```