PART 1

1)

a) You'd want a base 4 number system because it would take ½ the memory than a base 2 system would use.

As an interesting note, there did exist Decimal computers for a long while...

b) $33333_4 = 363_{10}$

2) A class is a blueprint for an object, saying how it should be constructed and what operations are possible for it. An object is one instance that a class is being used.

For example, all humans have basic attributes and things that they do. The average healthy human has skin, arms, and legs, amongst other things. The average healthy human can breathe, drink, eat, and learn, amongst other things. Humans do things that cells do. Humans also have someway of living, though Humans live differently from other organisms. So, you can make a “Human class” --

```
public class Human extends Cells implements Life{
        Boolean alive;
        String name;
        Hair hair;
        String gender;
        public Human(String name, String gender){
                this.name = name;
                this.gender = gender;
                // More things for being human...
        }
        public void setHair(Hair hair){
                this.hair = hair;
        }
        public void eat(){
                // Some functionality from Life
        }
        public void sleep(){
                // Somehow sleeps...
        }
        public void think(){
                // Somehow thinks...
        }
        public boolean isAlive(){
                return this.alive;
        }
 }
```

However, all Humans are different... that is, there exists different instances of the class Human. Therefore, there exist many different Human objects.

```
public static void main(String[] args){
 Human samantha = new Human("Samantha", "girl");
 samantha.setHair(new Hair("blonde"));
 Human john = new Human("John", "boy");
}
```

tl;dr : Classes are a blueprint for how you can make any object. Objects are a particular, practical version of your class.

3)

`12 / 5`	float	2.4
`“happy” + “3”`	String	happy3
`“happy” + 3`	String	happy3
`p1.equals(p1)`	boolean	true
`p1 == p1`	boolean	true
`int p = 3; p++`	int	4
`p1.toString().length()`	int	Length of the String version of p1

4)

a)

“([\\w]+ [\\w]+)([\\s]+)([\\d]{3})([\\s]+)”

Expression grouped by subpart.

b)

`“(\\w*)12.4+\\$$”`

Hello1234	No
111234$	Yes
Hello 1234$	Yes
Hello123$	No
12.4$	Yes

5)

Output:

ctor 1

ctor 2

intVal: 5

Address: Cherry Lane

intVal: 5

Address: Cherry Lane

intVal: 3

Address: Pine Tree Lane

intVal: 3

Address: Pine Tree Lane

6)

a)

```
public interface Printable{
	public void print(Object o);
}
```

b)

```
public class Computer implements Printable{
	String operatingSystem;
	private int bits;
	public void print(Object o){
		Computer x = (Computer) o;
		System.out.println(x);
		System.out.println(x.getBits());
	}
	public int getBits(){
		return this.bits;
	}
	public String getOperatingSystem(){
		return this.operatingSystem;
	}
	public void setBits(int b){
		if(b != 32 && b != 64){
			throw new Exception("This isn't a correct number!");
		} else {
			this.bits = b;
		}
	}

	public Computer(String os, int bits){
		if(bits != 32 && bits != 64){
			throw new Exception("Wrong number of bits!");
		} else {
			this.operatingSystem = os;
			this.bits = bits;
		}
	}
	public String toString(){
		return this.bits + "-bit " + this.operatingSystem;
	}
}
```

PART 2

If I have the following statements in my code,
`int x = 0;`
`return y / x;`
 	
The Java compiler will report **`DivideByZeroException`** because it knows that the value of x is zero in the return statement.
	False

Public mutable attributes of classes violate the object-oriented principle of information hiding	True
An object in memory can have, at most, two aliases	False. You can have “infinitely” many aliases... aka, the same object can have many names.
A switch statement can only have integers or characters as cases	False
I can leave off an import statement for **`java.util.*`** if I always refer to **`ArrayList`**s in my code as **`java.util.ArrayList`** instead of just **`ArrayList`**	True.
Code that is compiled is usually slower than code that is interpreted, which is why a lot of languages prefer interpreters to compilers	False.
I always write my name on exams	Do you?