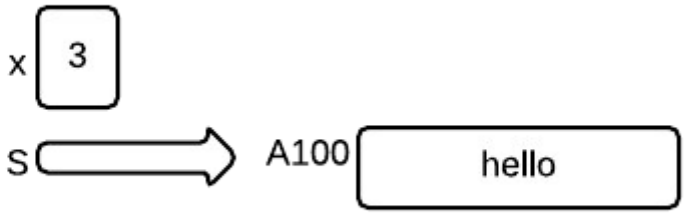
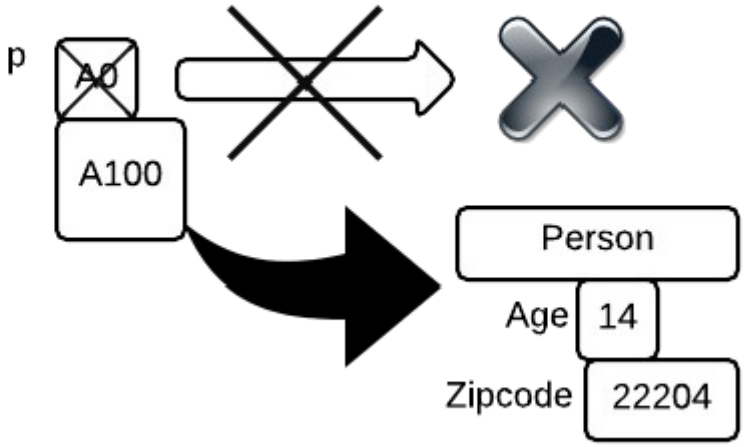
PART 1

1)

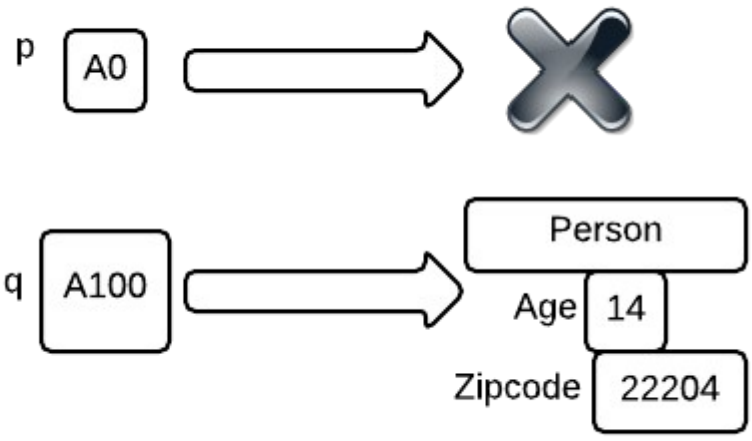
a)


x
3
S
A100
hello

b)


p
A0
A100
Person
Age
14
Zipcode
22204

c)


p
A0
q
A100
Person
Age
14
Zipcode
22204

2)

`12 / 5.0`	float	2.4
`'5' + “3”`	String	“53”
`5 % 3`	int	2
`String x = “hello”;` `x.toString();`	String	“hello”
`5 + 3 / 1`	int	8
`false || true`	boolean	true
`“7” + 3`	String	“73”

3)

Assumes that Person toString is:

```
public String toString(){
        return name + " " + age;
}
```

Output:

```
8.0
Sally 23
Joe 11
Sally 23
7
Kim 6
Joe 11
Bob 23
Joe 11
Kim 6
```

4)

Annotated in MyClass.java. Line numbers notated by Test Line / Line in File

Line 1 / 1	Did not include class keyword
Line 4 / 5	Default constructor not available for Exam1 class
Line 6 / 8	ex1 is an Exam1 type. Cannot be assigned a String.
Line 9 / 11	**name** is a final variable. Cannot be changed.
Line 10 / 12	**s** is a non static variable. Cannot be assigned like that from a static method.
Line 11 / 13	ex3 doesn't exist.
Line 14 / 16	**age** is a private variable in Exam1. Cannot be directly accessed.

```
//public MyClass{ <-- No Class keyword
public class MyClass{
        private String s;
        public static void main(String[] args){
//              Exam1 ex1 = new Exam1(); <-- This ctor doesn't exist
                Exam1 ex1 = new Exam1(3); // Random value
                Exam1 ex2 = new Exam1(3);
//              ex1 = "Hello"; <-- String cannot be assigned to ex1

                ex1.foo();
```

```
//              ex1.name = "David"; <-- name is a final var.
//              s = "Joe"; <-- s is not static.
//              ex3.format();

                System.out.println(ex1.name);
//              System.out.println(ex1.age); <-- age is private
        }
}
```

5)

Car.java is available in the folder.

```
// Part g
public class Car{

        // Part a
        String model;
        int mileage;

        // Part b
        public Car(){
                this.mileage = 0;
                this.model = "NONE";
        }

        // Part c
        public Car(String model, int mileage){
                // Part f -- error check
                if(mileage >= 0 && mileage < 100000){
                        this.mileage = mileage;
                } else {
                // It doesn't say what to do in case the mileage doesn't fit.
                // I choose to make something up, since it doesn't matter.
                this.mileage = 0;
                }
                this.model = model
        }

        // Part d
        public String toString(){
                return this.model + "has mileage : " + this.mileage;
        }

        // Part e
        public void incMileage(){
                this.mileage++;
        }
}
```

PART 2

A method's arguments are garbage collected after the method completes	True. The arguments don't exist outside of the scope.
The short type is the smallest integer type we can have	False. Byte is the smallest integer type.
The Java compiler would flag an error where you tried to divide the integer 3 by zero.	False. 3 / 0 is a Runtime error.
The java.lang package is automatically imported for you in Java programs that you write	True.
A static method can reference class instance data	False. Static method can only reference static members and methods.
The toString() method can still be called on any object of any type, even if one is not written for that specific type.	True. All classes are technically the child of the Object class.
I always write my name of exams.	Do you?