OOP (Classes & Objects) Practice Problems

# True/False Questions

1. Encapsulation is enforced by setting class attributes to private.
2. Coding a class creates a new simple data type.
3. Methods should be marked private and attributes should be marked public.
4. Constructors do not have a return type declared in the header.

# Multiple Choice Questions

1. An object’s state refers to:
   1. Its set of methods
   2. Its set of data fields with the current values
   3. Its skeleton
   4. Its constructor’s parameters
2. Class variables are also called:
   1. Accessors
   2. Constructors
   3. Attributes
   4. Behavior
3. Applying which term to a method allows the method to be called without creating an instance of the class it is in:
   1. Static
   2. Public
   3. Private
   4. Scope
4. Creating multiple methods (or constructors) with the same name but different parameters (number of parameters and/or data type of parameters and/or order of parameters) is called:
   1. Overriding
   2. Objections
   3. Overloading
   4. Overwriting
5. Visibility (public versus private) in a class is important because
   1. It allows overloading methods.
   2. It confuses students.
   3. It allows creating new objects from a class.
   4. It enforces encapsulation.

# Programming & Problem-Solving Questions

1. Almost all programming textbooks insist on including a Rectangle class example that includes two class variables - width and height. It also includes three methods inside the class – a constructor (that takes in a width and height as parameters), Area (that returns the area of the rectangle), and Perimeter (which returns the perimeter of the rectangle). Write class Rectangle, including its class variables and methods, and try not to fall asleep. Must… stay… awake….zzzzzzz
2. When people think of Rome, Italy, they think of its famous ancient ruins. However, less people know that the ruins are overrun with the “Gatti di Roma” – or Cats of Rome. Yes, that’s a real thing! The Italian government wants you to write a program to help track the cats, which starts with writing class Cat. All cats have a name as well as whether or not they have been seen by a doctor. By default, cats have not been seen by a doctor. Write the Cat class below, the attributes/variables for the class, as well as a constructor. Finally, write method “VisitDoctor” which sets the class variable to TRUE.
3. The government is working on a TOP SECRET project and needs your help. They want you to prototype a SpyTurtle! They need to know where the turtle is at all times, so each turtle has an X and Y coordinate. The turtle can hold up to 10 microphones and has the ability to drop a microphone, decreasing the number of microphones by 1. For this question, you will write a class, class variables, and a constructor for the SpyTurtle. You will also write a method called “DropMic” that decreases the number of microphones by 1.
4. From the year 2000-2014, "Bark Beetles" devasted the western part of the United States, eating 46 million of our country's 850 million acres of forested land. As such, the National Parks Service is reaching out for your help to develop a simulation that keeps track of the state of the trees. Design class tree that can keep track of the number of holes in the tree, the kind of tree (e.g. "Spruce") and the latitude/longitude (both floats) of the tree. By default, the tree has no holes. The class should also have the ability to "add" holes and "remove" holes, which increases/decreases the number of holes. If the tree has over 50 holes, it's considered to be "infested". Write a constructor and a method that returns whether or not the tree is infested.
5. Food Trucks are now “a thing” and are very profitable - making anywhere from $250,000-$500,000 per year! For this question, you’re going to write class TacoTruck that focuses on inventory. All taco trucks have a certain amount of meat, cheese and taco shells. It also knows how many tacos are sold (at $7 each!) Because your taco truck is known for the crazy amount of cheese it puts on a taco, for every taco you sell, two “units” of cheese are used (but only one unit of meat and one shell). Write class TacoTruck that has a method “sellTaco” that updates the inventory and also updates the number of tacos sold. Further, include a method that returns the total revenue the truck has made.
6. When you buy an air conditioner from a house, they're measured in "tons". If you buy too many tons, your house stays humid (and actually wears out faster); if you buy too few, your house won't cool down. Though it's an approximation, you'll need 1 ton of air conditioning for every 400 square feet in your house. Air conditioners are either on or off. Write class AirConditioner that has an amount of tons, a temperature, and whether or not it is running. Include a constructor that only takes in the square footage of a house and uses that to calculate the tons of the air conditioner. Then, write methods "increaseTemperature" and "decreaseTemperature" so the homeowner can control the temperature of their brand new AC unit.
7. If you aren't aware, there's are 5 massive amount of trash swirling in Pacific Ocean that are called "Garbage Patches". The largest, called "The Great Pacific Garbage Patch" that's now twice the size of Texas and up to 9 feet deep in places. There are an estimated 1,900,000 bits of plastic per square mile. For this question, you need to write class GarbagePatch, that will be measured in square miles. The class should include a method that returns the number of bits of piece of plastic, given its size in square miles. Do not forget include a constructor as, unfortunately, the garbage patches have to begin somehow.
8. Noside, LLC has contacted you to help them with a program for their company. Write a class based on the given Main program below.  The Class should contain the methods that match the way they are being called.  The Class should also have a constructor, a Boolean attribute that specifies whether the garden has been watered today, and a string that represents the currently planted crop.

BEGIN MAIN

Garden g1 = new Garden(true);

g1.plantCrop(“Watermelon”);

g1.waterCrops();

PRINT (g1.showValues());

END MAIN

1. Noside, LLC has contacted you to help them with a program for their company. Write a class based on the given Main program below.  The Class should contain the methods that match the way they are being called.  The Class should also have a constructor, a Integer attribute that specifies how many days in isolation the subject has endured, and a string that represents the subject’s feelings about being isolated.

BEGIN MAIN

IsolationCounter ic1 = new IsolationCounter(0);

ic1.incrementDays();

ic1.setMessageToWorld (“I’m ready to get back to school!”);

PRINT (ic1.showValues());

END MAIN

1. Noside, LLC has contacted you to help them with a program for their company. Write a class based on the given Main program below.  The Class should contain the methods that match the way they are being called.  The Class should also have a constructor, a Boolean attribute that specifies whether the light is on or off, and a string that represents the manufacturer of the light.

BEGIN MAIN

Light l1 = new Light(false);

l1.changeBrand(“Westinghouse”);

l1.turnOn();

PRINT (l1.showValues());

END MAIN

1. Imagine someone has written a Clock class for you with the following methods:

// Constructor

Clock (parameters: hours (integer), minutes (integer))

// Method1 – it increases the time on the clock by one hour

IncreaseHours (parameters:none)

// Method2 – it increases the time on the clock by the value passed in. This is different than Method 1

IncreaseMinutes (parameters: minutesToIncreaseBy (integer))

// Method 2 – prints out the hours, a colon :, and the minutes (e.g. 11:59)

DisplayTime (parameters: none)

// Declare two objects/variables of type Clock and initialize them correctly:

// Write the single statement to increase the number of minutes of the second clock by 5 minutes

// Write the two statements to make both Clocks display their time

// Write the output of making both Clocks display their time

1. Imagine someone has written a Dog class for you with the following methods:

// Constructor

Dog (parameters: name (string), weight (integer))

// Method1 – it makes the dog gain weight by a certain amount

eat (parameter: amount\_of\_food (integer))

// Method 2 – prints out the name of the dog, a comma, and its weight

bark (parameters: none)

// Declare two objects/variables of type Dog and initialize them correctly:

// Write the single statement to make the first Dog eat (an arbitrary amount of food)

// Write the two statements to make both Dogs bark

// Write the output of making both dogs bark here

1. Imagine someone has written a CoffeePot class for you with the following methods (all attributes are private):

// Constructor

CoffeePot(parameters: podPresent(Boolean), ouncesOfWater(int), brewTime(int))

// Method1 – it fills the water compartment to 100 ounces to allow brewing

fillWater (parameters:none)

// Method2 – it increases the brewing time of the coffee by the value sent in.

IncreaseBrewStrength (parameters: timeInSeconds (int))

// Method 3 – Brews coffee if a podPresent == true, brewTime > 0 and ouncesOfWater > 0.

BrewCoffee (parameters: none)

**Keurig has asked you to help them by creating two CoffeePot Objects so they can test their new Keuring brewing machine before releasing it for Holiday sales. Now you must call each method for the objects one time each in order (method 1, method 2 then method 3)**

1. Imagine someone has written a Printer class for you with the following methods (all attributes are private):

// Constructor

Printer(parameters: tonerPresent(Boolean), amountPaper(int), dataIn(Boolean)

// Method1 – turns fuser on to allow printing

turnFuserOn (parameters:none)

// Method2 – it increases the amount of paper by amount sent in

PCLoadLetter (parameters: paperAdded (int))

// Method 3 – prints out the print job if tonerPresent == true, amountPaper >= 1 and dataIn == true

PrintJob (parameters: none)

**Epson has asked you to help them by creating two Printer Objects so they can test their new printer designs before shipping them to stores. Now you must call each method for the objects one time each in order (method 1, method 2 then method 3)**

1. Imagine someone has written a CPU class for you with the following methods (all attributes are private):

// Constructor

CPU(parameters: speedGHz(double), numberOfCores(int), turboPresent(Boolean))

// Method1 – boots CPU and checks for issues

turnCPUOn (parameters:none)

// Method2 – it increases the speed by the amount sent in

overClockCPU (parameters: speedBoost (double))

// Method 3 – prints out the specifications for the CPU (speed in GHz, number of cores and if Turbo is present) on separate lines with descriptions for each spec.

printCPUSpecs (parameters: none)

**AMD has asked you to help them by creating two CPU Objects so they can test their new processors designs before announcing them to the world. Now you must call each method for the objects one time each in order (method 1, method 2 then method 3)**

1. Imagine someone has written an Owl class for you with the following methods (all attributes are private):

// Constructor

Owl(parameters: wingspan (double), ageInMonths(int), name(String))

// Method1 – it increases the age of the owl by one month

IncreaseAge (parameters:none)

// Method2 – it increases the wingspan of the owl by the value sent in.

IncreaseWingspan (parameters: inchesToAdd (double))

// Method 3 – prints out the owl’s name, its wingspan, its age and “Hooty hoo!” on separate lines with descriptions of the printed information.

DisplayOwl (parameters: none)

**KSU Athletics has asked you to help them by creating two Owl Objects so they can track their new owls for sporting events. Now you must call each method for the objects one time each in order (method 1, method 2 then method 3)**

1. KSU's Athletic Department has asked you to help them track their teams’ records for the season by creating a SportsTeam class with Sport, Wins and Losses attributes.
   1. First they need the Class Declaration & Attributes (all attributes should enforce encapsulation).
   2. They need a default constructor that sets all values to null for Strings and characters, zero (0) for numbers and false for Booleans.
   3. An Overloaded Constructor that sets attributes according to the values sent as parameters (should include values for all three attributes) to constructor is needed now.
   4. They need a Getter and a Setter for one attribute of your choice, getter should not have parameters and setter should not return a value.
   5. Class Method should calculate and print out the team’s win/loss ratio as a percentage (wins divided by losses) along with their sport the team plays.
   6. Now you will create the driver/test program that uses the SportsTeam Class you created previously.  Make sure to write a complete program that
   7. Creates two objects from the new class you created - one using the default CONSTRUCTOR, one using the overloaded CONSTRUCTOR.
   8. Calls the SETTER method for each of the two objects created for part one
   9. Calls the GETTER method for each of the two objects created for part one and does something with the returned information
   10. Calls the CLASS METHOD for one of the objects created for part one properly.
2. Beta Air Lines is a new startup air travel company and needs your assistance with a Airplane class so they can track their newly purchased airplanes.
   1. First you need to declare the class and attributes, making sure that all attributes enforce encapsulation:
   2. Airplane – Passenger Limit, Destination, Tickets Sold
   3. Now Beta Air Lines needs a default constructor for the Airplane class that sets all values to null for Strings and characters, zero (0) for numbers and false for Booleans:
   4. Now create an Overloaded Constructor that sets the attributes according to the values sent as parameters (should include values for all three attributes) to constructor:
   5. BAL needs a Getter and a Setter for one attribute of your choice, getter should not have parameters and setter should not return a value:
   6. Airplane Class Method needed should calculate and print out how many (if any) passengers have to be bumped due to the flight being overbooked (Tickets sold – Passenger Limit).  This is a common practice amongst airlines so do not be surprised that they oversold tickets for a flight!
   7. Now you will create the driver/test program that uses the Airplane Class you created previously.  Make sure to write a complete program that
   8. Creates two objects from the new class you created - one using the default CONSTRUCTOR, one using the overloaded CONSTRUCTOR.
   9. Calls the SETTER method for each of the two objects created for part one
   10. Calls the GETTER method for each of the two objects created for part one and does something with the returned information
   11. Calls the CLASS METHOD for one of the objects created for part one properly.
3. The Whatever Channel (named because the weather in 2020) is a newly created competitor to the well known Weather Channel and needs your assistance with getting started by creating a Weather Tracker class for them with the following attributes: Name of storm, type of storm and direction. Make sure all attributes enforce encapsulation.
   1. Now write a Default Constructor that sets all values to null for Strings and characters, zero (0) for numbers and false for Booleans.
   2. Here you should write an Overloaded Constructor that sets attributes according to the values sent as parameters (should include values for all three attributes) to constructor.
   3. Now they need a Getter and a Setter for one attribute of your choice, getter should not have parameters and setter should not return a value:
   4. Last they need a Class Method that prints the name of the storm, the type of the storm and the direction that the storm is headed.
   5. Now you will create the driver/test program that uses the Weather Tracker Class you created previously.  Make sure to write a complete program that
   6. Creates two objects from the new class you created - one using the default CONSTRUCTOR, one using the overloaded CONSTRUCTOR.
   7. Calls the SETTER method for each of the two objects created for part one
   8. Calls the GETTER method for each of the two objects created for part one and does something with the returned information
   9. Calls the CLASS METHOD for one of the objects created for part one properly.
4. KSU's Botany Department has asked for your assistance in creating a Plant Tracker class with attributes for toxicity, medicinal use and color to help their undergraduate students catalog and avoid any dangerous plants they encounter on campus.  Make sure that all attributes enforce encapsulation.
   1. Here write a Default Constructor that sets all values to null for Strings and characters, zero (0) for numbers and false for Booleans.
   2. Next create an Overloaded Constructor that sets attributes according to the values sent as parameters (should include values for all three attributes) to constructor.
   3. They need a Getter and a Setter for one attribute of your choice, getter should not have parameters and setter should not return a value:
   4. Lastly a Class Method is necessary, it should print out if a plant is toxic, has medicinal properties or both with appropriate statements.
   5. Now you will create the driver/test program that uses the Plant Tracker Class you created previously.  Make sure to write a complete program that
   6. Creates two objects from the new class you created - one using the default CONSTRUCTOR, one using the overloaded CONSTRUCTOR.
   7. Calls the SETTER method for each of the two objects created for part one
   8. Calls the GETTER method for each of the two objects created for part one and does something with the returned information
   9. Calls the CLASS METHOD for one of the objects created for part one properly.
5. Write class GameConsole based on the given Main program below.  The class must have a constructor, a Boolean attribute that specifies whether the game console is on or off, and a string attribute that represents the game currently loaded in the console. Finally, the class should have methods turnOn( ), loadGame( ), and gameStart( ) that would work with (i.e. compile with) the main program below.

BEGIN MAIN

GameConsole gc1 = new GameConsole(false, “ “); /\* You must have a constructor that takes in a Boolean and a string and initializes the attributes to those values\*/

gc1.turnOn(); // Changes the state of the game console to on

gc1.loadGame(“Thermonuclear War”); // Changes the name of the currently loaded game

PRINT(gc1.gameStart()); // Prints out the name of the currently loaded game

END MAIN

1. Write class Weedeater based on the given Main program below.  The class must have a constructor, a Boolean attribute that specifies whether the weed eater is running or stopped, and an integer that represents the amount of weed line/string left. Finally, the class should have a start() method, an addLine() method and a showLineLength() method that would work with (i.e. compile with) the main program below.

BEGIN MAIN

WeedEater we1 = new WeedEater(true, 2); /\* You must have a constructor that takes in a Boolean and an integer and initializes the attributes to those values \*/

we.start(); // Changes the state of the weed eater to running

we.addLine(5); // Changes the state by adding 5 feet of line to the weed eater

PRINT(we1.showLineLength()); //prints out the current amount of fishing line left

END MAIN

1. Write class KitchenMixer based on the given Main program below.  The class must have a constructor, a String attribute that specifies what mixing blade is attached, and an integer that represents the speed of the mixer. Finally, the class should have a setSpeed() method, a changeBlade() method and a showSettings() method that would work with (i.e. compile with) the main program below.

BEGIN MAIN

KitchenMixer km1 = new KitchenMixer(“Dough hook”, 2); /\* You must have a constructor that takes in a string and integer and initializes the attributes to those \*/

km1.setSpeed(2); // Change the state of the kitchen mixer to a speed of 2

km1.changeBlade(“Whisk”); // Update the kind of mixing blade attached

PRINT(km1.showSettings()); //prints out the current mixing blade and speed

END MAIN

1. Kenmore has contacted you to help write code to control their new line of home appliances. Write the “Refrigerator” class that has tracks the temperature of the refrigerator and the amount of water remaining in the water dispenser (max is 100 ounces). Kenmore also needs you to create a constructor and methods to increase or decrease the temperature along with dispensing water (8 ounces at a time) and refilling the water tank from a connected water hose.
2. GDOT has contacted you to help write code to control the traffic signals in Georgia. You must create a Traffic Signal class with three hidden attributes (green, yellow and red), two constructors (a default that sets all lights to on and an overloaded that sets only green to on for testing purposes) and a method that changes the active light to the next in the sequence (green – yellow – red). Now you must create two Traffic Signal objects from the class you just created to prove that your class works properly for GDOT, the first object should use the default constructor and the second object should use the overloaded constructor. Using the objects you just created, call the method to change the light to the next in the sequence.
3. GDOT has contacted you to help write code to control the cross walk signals in Georgia. You must create a Crosswalk Signal class with three hidden attributes (Walk, Hurry and Wait), two constructors (a default that sets all lights to on and an overloaded that sets Hurry to on for testing purposes) and a method that changes the active light to the next in the sequence (Walk, – Hurry – Wait). Now you must create two Crosswalk Signal objects from the class you just created to prove that your class works properly for GDOT, the first object should use the default constructor and the second object should use the overloaded constructor. Using the objects you just created, call the method to change the light to the next in the sequence.
4. GDOT has contacted you to help write code to control the railroad signals (barriers with flashing lights and sounds across railroad tracks) in Georgia. You must create a Railroad Signal class with three hidden attributes (Up, Down and Signal), two constructors (a default that sets Up to on, Down and signal to off, and an overloaded that sets all Up to off, Down and Signal to on) and a method that changes the position and activates or deactivates the Signal. Now you must create two Railroad Signal objects from the class you just created to prove that your class works properly for GDOT, the first object should use the default constructor and the second object should use the overloaded constructor. Using the objects you just created, call the method to change the lights and arms to the next in the sequence.
5. A PMC (private military contractor) has contacted you to help write code to control the Security Checkpoints for their company. You must create a Checkpoint class with a hidden attributes (Authorized, Guest, Unauthorized), two constructors (a default that sets all access levels to false and an overloaded that sets Authorized and Unauthorized to false and Guest to true for testing purposes) and a method that changes the access level to the next in the sequence (Authorized – Guest – Unauthorized). Now you must create two Checkpoint objects from the class you just created to prove that your class works properly for the PMC, the first object should use the default constructor and the second object should use the overloaded constructor. **)** Using the objects you just created, call the method to change the access level to the next in the sequence.
6. GDOT has contacted you to help write code to control the Parking Lot barricades (the arms that come down to prevent cars from entering without paying) in Georgia. You must create a Parking Lot class with three hidden attributes (Paid, Up and Down), two constructors (a default that sets Paid and Up to false, Down to true and an overloaded that sets Paid and Up to true and Down to false for testing purposes) and a method that changes the Paid to the opposite value and changes Up and Down appropriately. Now you must create two Parking Lot objects from the class you just created to prove that your class works properly for GDOT, the first object should use the default constructor and the second object should use the overloaded constructor.Using the objects you just created, call the method to change Paid to the opposite value and to raise or lower the arms accordingly.
7. Please complete the Hurricane class outlined below.

CLASS Hurricane {

**//Attribute(s)**

CREATE PRIVATE category; //a whole number between 1 and 5

**//Default Constructor**

Hurricane (parameters:  none) {

category = 1;

}

}

* 1. Create an overloaded constructor that takes in a single parameter for category.
  2. Create a getter method for the category attribute.
  3. Create a setter method for the category attribute.
  4. According to the National Hurricane Center (<https://www.nhc.noaa.gov/aboutsshws.php>):
  5. While all hurricanes produce life-threatening winds, hurricanes rated Category 3 and higher are known as major hurricanes.  Create a method that uses the value of the category attribute to determine whether or not the storm qualifies as a major storm.  If the storm meets the requirements for a major storm, the method should print the following message:

WARNING:  This is a major storm!  Seek shelter immediately!

* 1. In a complete program, separate from the CLASS file, you need to
  2. Create an object of the Hurricane class that represents a category 4 storm.
  3. The hurricane has increased in intensity.  Please make a method call to adjust the storm’s category to category 5.

1. Please complete the Typhoon class outlined below.

CLASS Typhoon {

**//Attribute(s)**

CREATE windspeed;//in mph

**//Default Constructor**

Typhoon (parameters:  none) {

windspeed = 74;

}

}

* 1. Create an overloaded constructor that takes in a single parameter for windspeed.
  2. Create a getter method for the windspeed attribute.
  3. Create a setter method for the windspeed attribute.
  4. According to the National Hurricane Center (https://www.nhc.noaa.gov/aboutsshws.php):
  5. In the western North Pacific, the term "super typhoon" is used for tropical cyclones with sustained winds exceeding 150 mph.
  6. Create a method that uses value of the windspeed attribute to determine whether or not a particular typhoon qualifies as a super typhoon.  This method should RETURN a true/false value.
  7. In a complete program, separate from the CLASS file, you need to
  8. Create an object of the Typhoon class with windspeeds of 140 mph.
  9. The Typhoon has increased in intensity.  Please make a method call to adjust the storm’s windspeed to 155 mph.

1. Please complete the Precipitation class outlined below.

CLASS Precipitation {

**//Attribute(s)**

CREATE rainfall;//in inches

**//Default Constructor**

Precipitation (parameters:  none) {

rainfall = 140;}

}

* 1. Create an overloaded constructor that takes in a single parameter for rainfall.
  2. Create a getter method for the rainfall attribute.
  3. Create a setter method for the rainfall attribute.
  4. Located on the Olympic Peninsula of Washington State, the Hoh Rainforest is one of the rainiest places in the United States, averaging 140 inches of rain per season.
  5. Create a method that uses value of the rainfall attribute to determine whether or not a particular rainy season is below average.
  6. If the rainfall is lower than average, the method should print the following message:  “There has been less rainfall than average this season.”
  7. In a complete program, separate from the CLASS file, you need to
  8. Create an object of the Precipitation class with rainfall of 90 inches.
  9. Please make a method call to adjust the object’s rainfall to 200 inches.

1. Please complete the Blizzard class outlined below.

CLASS Blizzard {

**//Attribute(s)**

CREATE visibility;//in miles

CREATE windspeed; //in mph

**//Default Constructor**

Blizzard (parameters:  none){

visibility = .25;

}

}

* 1. Create an overloaded constructor that takes in visibility and assigns a value of 35 to the windspeed attribute.
  2. Create a getter method for the visibility attribute.
  3. Create a setter method for the visibility attribute.
  4. The National Weather service defines a storm as a blizzard if winds exceed 34 mph and visibility is less than 0.25 miles.
  5. Create a method that uses value of the visibility and windspeed attributes to determine whether or not a storm qualifies as a blizzard.
  6. This method should RETURN a true/false value.
  7. In a complete program, separate from the CLASS file, you need to
  8. Create an object of the Blizzard class with visibility of 25 miles.
  9. Please make a method call to adjust the object’s visibility to 50 mph.

**Junkyard Wars** - You are participating on a gameshow where your team must build a functioning car from junkyard scrap. You are tasked with creating a class for one of the car’s components.

1. Create an **Odometer** Class with a secure **Mileage** attribute.
2. Create a default Constructor with no parameters that assigns a default value for the variable.
3. Create an overloaded Constructor (with a parameter for the variable).
4. Create two methods (one that increases current mileage reading by 1 and a second method that returns current value of mileage attribute). NOTE: these methods must not print anything.
5. Create a new object of the class you just created using the overloaded constructor (send in any value you choose).
6. For your new object, call the method that increases the attribute then print out the attribute’s current value by calling the method that returns its value

**Junkyard Wars** - You are participating on a gameshow where your team must build a functioning car from junkyard scrap. You are tasked with creating a class for one of the car’s components.

1. Create a **Tachometer** Class with a secure **RPM** attribute.
2. Create a default Constructor with no parameters that assigns a default value for the variable.
3. Create an overloaded Constructor (with a parameter for the variable).
4. Create two methods (one that increases current RPM reading by 1 and a second method that returns current value of RPM attribute). NOTE: these methods must not print anything.
5. Create a new object of the class you just created using the overloaded constructor (send in any value you choose).
6. For your new object, call the method that increases the attribute then print out the attribute’s current value by calling the method that returns its value

**Junkyard Wars** - You are participating on a gameshow where your team must build a functioning car from junkyard scrap. You are tasked with creating a class for one of the car’s components.

1. Create a **Speedometer** Class with a secure **MPH** attribute.
2. Create a default Constructor with no parameters that assigns a default value for the variable.
3. Create an overloaded Constructor (with a parameter for the variable).
4. Create two methods (one that increases current MPH reading by 1 and a second method that returns current value of MPH attribute). NOTE: these methods must not print anything.
5. Create a new object of the class you just created using the overloaded constructor (send in any value you choose).
6. For your new object, call the method that increases the attribute then print out the attribute’s current value by calling the method that returns its value

**Junkyard Wars** - You are participating on a gameshow where your team must build a functioning car from junkyard scrap. You are tasked with creating a class for one of the car’s components.

1. Create a **Windows** Class with a secure **Position** attribute.
2. Create a default Constructor with no parameters that assigns a default value for the variable.
3. Create an overloaded Constructor (with a parameter for the variable).
4. Create two methods (one that changes the value (up/down) and a second method that returns the current status of window attribute). NOTE: these methods must not print anything.
5. Create a new object of the class you just created using the overloaded constructor (send in any value you choose).
6. For your new object, call the method that increases the attribute then print out the attribute’s current value by calling the method that returns its value

**Junkyard Wars** - You are participating on a gameshow where your team must build a functioning car from junkyard scrap. You are tasked with creating a class for one of the car’s components.

1. Create a **DoorLocks** Class with a secure **Locked** attribute.
2. Create a default Constructor with no parameters that assigns a default value for the variable.
3. Create an overloaded Constructor (with a parameter for the variable).
4. Create two methods (one that changes the current value to the other (locked/unlocked) and a second method that returns the current status of Locked attribute). NOTE: these methods must not print anything.
5. Create a new object of the class you just created using the overloaded constructor (send in any value you choose).
6. For your new object, call the method that increases the attribute then print out the attribute’s current value by calling the method that returns its value

**Junkyard Wars** - You are participating on a gameshow where your team must build a functioning car from junkyard scrap. You are tasked with creating a class for one of the car’s components.

1. Create a **ComfortControl** Class with a secure **temperature** attribute.
2. Create a default Constructor with no parameters that assigns a default value for the attribute.
3. Create an overloaded Constructor (with a parameter for the attribute).
4. Create two methods (one that increases current temperature reading by 1 degree and a second method that returns current value of temperature attribute). NOTE: these methods must not print anything.
5. Create a new object of the class you just created using the overloaded constructor (send in any value you choose).
6. For your new object, call the method that increases the attribute then print out the attribute’s current value by calling the method that returns its value

**Junkyard Wars** - You are participating on a gameshow where your team must build a functioning car from junkyard scrap. You are tasked with creating a class for one of the car’s components.

1. Create a **ComfortControl** Class with a secure **temperature** attribute.
2. Create a default Constructor with no parameters that assigns a default value for the attribute.
3. Create an overloaded Constructor (with a parameter for the attribute).
4. Create two methods (one that increases current temperature reading by 1 degree and a second method that returns current value of temperature attribute). NOTE: these methods must not print anything.
5. Create a new object of the class you just created using the overloaded constructor (send in any value you choose).
6. For your new object, call the method that increases the attribute then print out the attribute’s current value by calling the method that returns its value

Write classes based on the given requirements below.  The classes must have a constructor, secure attributes and methods that match what is described.

**Class** – PaperShredder

**Attributes**: AmountOfShreddedPaper

**Methods**: EmptyBin (reduces amount to shredded paper to 0 and returns the amount)

Forward (prints a statement that the paper shredder is running in forward direction)

Reverse (prints a statement the the paper shredder is running in reverse to clear an overload)

Write a complete MAIN program that creates an object of the class created in the previous question and calls each method for the object properly.

**Class** – GarbageDisposal

**Attributes**: WaterOn

**Methods**: NeedsCleaning (returnsa value indicated if the garbage disposal needs cleaning or not)

ClearJam (prints a statement that a jam has been cleared)

Run (prints a statement indicating it is running if the water is on, otherwise it jams)

Write a complete MAIN program that creates an object of the class created in the previous question and calls each method for the object properly.

**Class** – Copier

**Attributes**: AmountOfPaper

**Methods**: CopyPapers (prints a statement that the copier is copying a number of pages sent is as a parameters)

HolePunch (prints a statement that papers have been holepunched)

Staple (returns a value to indicate if the papers have been stapled or not)

Write a complete MAIN program that creates an object of the class created in the previous question and calls each method for the object properly.

**Class** – TrashCompactor

**Attributes**: AmountOfTrash

**Methods**: CompactTrash (reduces the amount of trash in the compactor by half)

ClearJam (prints a statement that the jam has been cleared)

NeedsCleaning (returns a value indicating if the trash compactor needs to be cleaned)

Write a complete MAIN program that creates an object of the class created in the previous question and calls each method for the object properly.

A knock off electronics company has contracted you to help them write the software for their new line of video games.

Sorny needs you to write a **PirateShip** class that includes:

* Class Declaration & Attributes: int Crew and int Treasure
* Default Constructor that sets Crew to 30 and Treasure to 20
* Overloaded Constructor that sets attributes according to the values sent as parameters to constructor
* Two Methods – loadTreasure (loads treasure onto ship as long as amount of treasure is not greater than crew \* 10) and AbandonShip (sets crew to zero and treasure to zero). Neither method should return a value.
* Create two objects from the new class you created - one using the default Constructor, one using the overloaded Constructor.
* Call both methods for each of the two objects created in question 12 **(10 points)**

A knock off electronics company has contracted you to help them write the software for their new line of televisions.

Sorny needs you to write a **Television** class that includes:

* Class Declaration & Attributes: Boolean Power and int Channel
* Default Constructor that sets power to off and channel to 3
* Overloaded Constructor that sets attributes according to the values sent as parameters to constructor
* Two Methods – ChangeChannel (sets channel attribute to value sent in as a parameter) and ChangePowerMode (sets power to opposite of current value). Neither method should return a value.
* Create two objects from the new class you created - one using the default Constructor, one using the overloaded Constructor.
* – Call both methods for each of the two objects created in question 12 **(10 points)**

A knock off robotics company has contracted you to help them write the software for their new line of space shuttles.

Edison needs you to write a **Shuttle** class that includes:

* Class Declaration & Attributes: String Name and int CrewSize
* Default Constructor that sets Name to “SpaceY” and crew to 0
* Overloaded Constructor that sets attributes according to the values sent as parameters to constructor
* Two Methods – changeName (sets name attribute to value sent in as a parameter) and PrepCrew (sets number of crew equal to number of letters in shuttle name). Neither method should return a value.
* Create two objects from the new class you created - one using the default Constructor, one using the overloaded Constructor.
* Call both methods for each of the two objects created in question 12 **(10 points)**

Kenmore has contacted you to help them create a class for their newest washing machine. Write the “WashingMachine” class that has the following specifications:

Attributes (should be invisible to users):

1. Amount of water (number)
2. Minutes left (the number of minutes left in the cycle)
3. Water level (number indicating how many quarts of water are in the machine

Methods:

1. Constructors - default and overloaded for all attributes
2. Add water - increments water level by 1
3. Drain water - removes all water
4. Spin – decreases the minutes left by 0.01
5. getTime – returns number of minutes left
6. setTime – sets the number of minutes to run

Write class Thermostat to the following specifications:

Attributes (should be invisible only inside the class): heatOn (Boolean), and temp (number)

Methods:

* Constructors (default and overloaded for all attributes)
* turnHeatOn(), turnHeatOff() – sets the heat on or off
* getTemp(), setTemp() – getter/setter for the temp attribute
* sense() – turns the heat on if it’s below 60 degrees

Write class *Clock* that has the following specifications:

Attributes (only visible inside the class): Seconds, Minutes, and Hours

Methods:

* Constructors (default and overloaded for all attributes)
* AddSecond() – adds one second to the time. Note: account for what happens at 60 seconds (e.g. you can’t have 61 seconds)
* AddMinute() – same as AddSecond(), but for minutes
* Accessors methods for getting minutes, hours, and seconds

RaceTrac® has contacted you to help them create a GasPump class for their newest gas station. Write class GasPump that has the following specifications:

Attributes (should not be visible outside of the class): Total amount pumped, amount left in tank, price per gallon

Constructors (2): default and overloaded for all attributes

Methods (6):

* RefillTank() – sets the amount left in the tank to “full” at 1000 gallons
* DispenseGas() – dispenses gas by one (1) gallon
* TotalSales() – calculates and returns how much money this pump has made
* setPrice(), getPrice(), getAmountInTank() – accessors and modifiers

iRobot has contacted you to help them create a Robot class for their newest automaton. Write class Robot that has the following specifications:

Attributes (should not be visible outside the class): Name, Speed, Latitude (North/South – a number), and Longitude (East/West – a number)

Constructors: default and overloaded for all attributes

Methods: North(), East(), West(), South() – which increment/decrement either the latitude or longitude by the current speed of the robot (e.g. North() will change the latitude of the robot)

SetSpeed(), getSpeed() – sets/gets the current speed of the robot

**Storage Wars:** Create classes with the names and variables/attributes listed below. Create a default Constructor with no parameters that assigns default values for each variable. Then, create an overloaded Constructor (with one parameter for each variable). Finally, create a method that returns the total number of items in the storage container. NOTE: this method must not print anything.

**Piggybank – quarters, dimes, nickels, pennies**

**Trunk – baseballs, softballs, basketballs, footballs**

**Suit case – pants, shirts, socks, underwear**

**Backpack – science books, math books, literature books, social sciences books**

**Locker – textbooks, pencils, pens, notebooks**

**Treasure Chest – diamonds, emeralds, sapphires and rubies**

Create a driver program and two new objects of the class you just created, one using the default constructor, one using the overloaded constructor. For each of the two objects, call the method that calculates the total number of items in each object. Then, print out the values the method calls return to the screen.

**Ocean Exploration** – You have been tasked by NOAA (National Oceanic and Atmospheric Administration) with helping write the code for their new salvage diving submersible **(50 points)**. They require:

1. A **Submersible** Class with secure **Depth, Latitude** and **Longitude** attributes.
2. A default Constructor with no parameters that assigns default values for the attributes.
3. A method that prints out the current Latitude & Longitude of the submersible
4. Getter & Setter methods for the Depth attribute. NOTE: these methods must not print anything.
5. Create two new objects of the class you just created using constructor.
6. For your new objects, call the methods in the following order: **(20 points)**
7. Set method for each object’s depth:
   1. Depth of 25 feet for first object
   2. Depth of 75 feet for second object
8. Print method for each object along with the get method for each object’s depth

**Space Exploration** – You have been tasked by NASA (National Aeronautics and Space Administration) with helping write the code for their new deep space probe **(50 points)**. They require:

1. A **Probe** Class with secure **Destination, Distance** and **Fuel** attributes.
2. A default Constructor with no parameters that assigns default values for the attributes.
3. A method that prints out the current Distance & Fuel of the probe
4. Getter & Setter methods for the Altitude attribute. NOTE: these methods must not print anything.
5. Create two new objects of the class you just created using constructor.
6. For your new objects, call the methods in the following order:
7. Set method for each object’s destination:
   1. Destination of Jupiter for first object
   2. Destination of Saturn for second object
8. Print method for each object along with the get method for each object’s altitude.