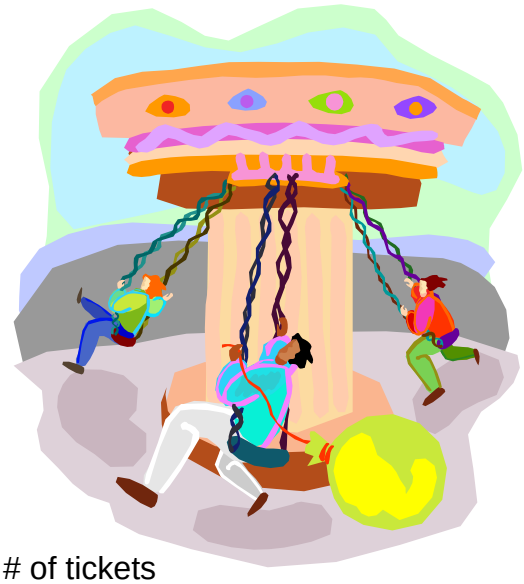
COMP1406 - Assignment #2

(Due: **Monday, January 27th @ 11:30pm**)

In this assignment, you will create 3 objects and get them to interact together. Specifically, you will create a **TicketBooth** object in which **Person** objects will purchase tickets in order to get onto **Rides.** You will define these classes, then do testing which simulates many people buying tickets and getting on the rides. The testing will then compute additional information about the rides and ticket booth.

(1) The **Ride** Class



Define a class called **Ride** that defines the following attributes:

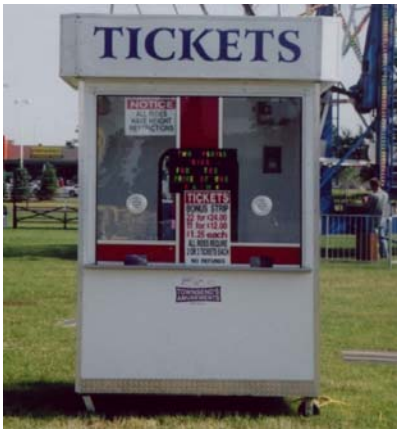
- name - a **string** indicating the name of the ride
- ticketsRequired - **int** indicating the number of tickets that must be used to get onto this ride
- heightRequirement - **float** indicating minimum height requirement for a person to be allowed on this ride
- numberOfRiders - **int** indicating the number of riders on the ride so far

Write a constructor that takes as parameters the name of the ride, # of tickets required and height requirement. Also, write a zero-parameter constructor. Lastly, write a **toString()** method that returns a string in the following format:

```
Roller Coaster requiring 6 tickets with a height restriction of 4.9'
```

NOTE: You DO NOT need to make private attributes, nor get/set methods for this assignment.
Write a test program called **RideTestProgram.java** that creates two rides using the 3-parameter constructor and one ride using the 0-parameter constructor and then displays the rides to the console. Run your code to make sure that it works.

(2) The **TicketBooth** Class



Define a class called **TicketBooth** that defines the following attributes:

- moneyMade - a **float** indicating the amount of money received by this booth so far
- availablePasses - an **int** indicating the number of passes that are available at this booth
- availableTickets - an **int** indicating the number of tickets that are available at this booth

Define the following two **static** constants in the **TicketBooth** class:

- TICKET_PRICE - the price for each individual ticket ... set it to be $0.50.
- PASS_PRICE - the price for a ride pass ... set it to be $16.50.

Write a zero-parameter constructor as well as one that takes an initial # of passes that the booth has available. Also, write a constructor that takes an initial number of passes that the booth has available as well as the number of tickets that it has available. So … you will write 3 constructors altogether. Lastly, write a **toString()** method that returns a string with the following format:

```
Ticket booth with 5 passes and 28 tickets
```

Making use of the two class constants above whenever possible, write these 2 methods:

- a **sellPass()** method that simulates the selling of a ride pass at this booth (*Hint*: Think of what happens in real life and then determine how the TicketBooth object changes as a result of this method. Make sure to handle the situation where there are no passes available).

- a **sellTickets(**int **num)** method that simulates the selling of the specified number of tickets at this booth. If there are not enough tickets to accommodate the requested amount then nothing happens, no selling takes place. (*Hint*: Think the same way as you did above).

Test your code with this program called **TicketBoothTestProgram.java:**

```
public class TicketBoothTestProgram {
    public static void main(String args[]) {
          TicketBooth       booth1, booth2;

          booth1 = new TicketBooth(5, 50);
          booth2 = new TicketBooth(1, 10);

          System.out.println("Here are the booths at the start:");
          System.out.println("   " + booth1);
          System.out.println("   " + booth2);

          // Simulate selling 2 passes, 5 tickets, 12 tickets and 3 tickets from booth1
          booth1.sellPass();
          booth1.sellPass();
          booth1.sellTickets(5);
          booth1.sellTickets(12);
          booth1.sellTickets(3);

          // Simulate selling 2 passes, 5 tickets, 12 tickets and 3 tickets from booth2
          booth2.sellPass();
          booth2.sellPass();
          booth2.sellTickets(5);
          booth2.sellTickets(12);
          booth2.sellTickets(3);

          // Make sure that it all worked
          System.out.println("\nBooth 1 has made $" + booth1.moneyMade);
          System.out.println("Booth 2 has made $" + booth2.moneyMade);
          System.out.println("\nHere are the booths at the end:");
          System.out.println("   " + booth1);
          System.out.println("   " + booth2);
    }
}
```

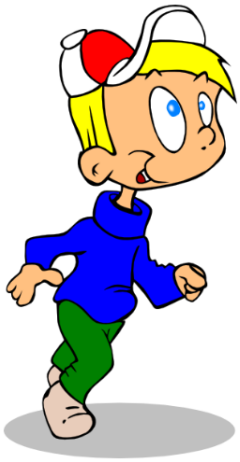
The code should produce the following output **exactly**:

```
Here are the booths at the start:
   Ticket booth with 5 passes and 50 tickets
   Ticket booth with 1 passes and 10 tickets

Booth 1 has made $43.0
Booth 2 has made $20.5

Here are the booths at the end:
   Ticket booth with 3 passes and 30 tickets
   Ticket booth with 0 passes and 2 tickets
```

(3) The **Person** Class



Define a class called **Person** which has attributes called **height** , **money** , **ticketCount** and **hasPass**.

The **money** and **height** variables are floats representing the amount of money the person currently has and the person's height (in inches).

The **ticketCount** is an integer indicating the number of tickets the person currently has and **hasPass** is a **boolean** indicating whether or not the person has a pass to get on the rides (assume a person can have at most 1 pass).

Write a constructor & all necessary methods in the **Person** class so that the following code produces the output as shown below:

```
public class PersonTester {
    public static void main(String args[]) {
        Person     mary;

        mary = new Person(4.9f, 20.00f);

        System.out.println(mary.height);
        System.out.println(mary.money);
        System.out.println(mary.ticketCount);
        System.out.println(mary.hasPass);
        System.out.println(mary);   // Notice the money is properly formatted

        mary.ticketCount = 3;
        System.out.println(mary);

        mary.useTickets(2);   // You have to write this method
        System.out.println(mary);

        mary.hasPass = true;
        System.out.println(mary);
    }
}
```

Here is the output that your program MUST produce:

```
4.9
20.0
0
```

```
false
4.9' person with $20.00 and 0 tickets
4.9' person with $20.00 and 3 tickets
4.9' person with $20.00 and 1 tickets
4.9' person with $20.00 and a pass
```

Make sure to test your class with the test code above BEFORE you continue.

(4) The FUN Stuff

Add the following instance methods to the **Person** class:

- **buyPass(TicketBooth booth)** - This method simulates the person buying a ride pass from the given booth object. It should modify the person and the booth in the appropriate way, and then return a **boolean** indicating whether or not the transaction was successful. The transaction is successful ONLY if the booth has a pass available AND the person has enough money for the pass.

- **buyTickets(**int **number, TicketBooth booth)** - This method simulates the person attempting to buy the specified number of tickets from the given booth object. It should modify the person and the booth in the appropriate way. If the booth does not have the specified **number** of tickets available, then the **number** of tickets to be sold should be reduced to the number of remaining tickets available. Then, if the person has enough money to buy the resulting number of tickets (i.e., either the requested amount or the reduced amount) then the purchase is made. If the person does not have enough money to buy the tickets, then the number of tickets sold should be further reduced to the number of tickets that the person can afford. The method must return an integer representing the number of tickets actually sold. Make sure that the booth maintains a proper total of all money paid to it.

- **allowedToRide(Ride aRide)** - This method determines whether or not the person is allowed to ride on the given ride. A person can get on a ride if he/she meets the ride's minimum height requirements AND the person either has a pass OR the required number of tickets for that ride. The method MUST return a **boolean** and MUST NOT have any System.out.println code within it. Also, it MUST be written efficiently ... you will lose marks for unnecessary code ... make use of any methods you already have.

- **getOn(Ride aRide)** - This method simulates the person getting on the given ride. A person can get on a ride ONLY if he/she is allowed to (i.e., make sure you use the method you just wrote). Make sure that the **Person** AND the **Ride** object are updated accordingly (no tickets are used if the person has a pass). The method **MUST NOT** have any **System.out.println** code within it and it should return **true** if the person was able to get on the ride and **false** otherwise.

Make sure to now test your code with the following test case in a class called **FairTester**:

```
public class FairTester {

    public static void main(String args[]) {
        Ride          coaster, ferris, merryGo, tosser;
        Person        billy, donna, fredy, harry, larry;
        TicketBooth   booth;

        // Make some rides on which people can ride, specify name, #tickets & height req
        coaster = new Ride("Roller Coaster", 6, 4.25f);
        ferris = new Ride("Ferris Wheel", 5, 3.1f);
        merryGo = new Ride("Merry-Go-Round", 2, 0);
        tosser = new Ride("Tummy Tosser", 7, 4.9f);

        // Make a booth, specifiying the number of available passes and tickets
        booth = new TicketBooth(4, 100);

        // Make some people by specifying their height and their money amounts
        billy = new Person(4.9f, 10.00f);
        donna = new Person(3.0f,  5.00f);
        fredy = new Person(6.0f,  0.00f);
        harry = new Person(4.8f, 78.50f);
        larry = new Person(4.0f, 50.00f);

        System.out.println(booth);

        System.out.println("\nBilly is a " + billy);
        System.out.println("Billy just bought " + billy.buyTickets(20, booth) +
                           " tickets.");
        System.out.println("Billy attempting to go on the " + coaster);
        System.out.println("Billy got on: " + billy.getOn(coaster));
        System.out.println("Billy attempting to go on the " + tosser);
        System.out.println("Billy got on: " + billy.getOn(tosser));
        System.out.println("Billy is now a " + billy + "\n");

        System.out.println("Donna is a " + donna);
        System.out.println("Donna is trying to buy a pass...was she successful: "
                           + donna.buyPass(booth));
        System.out.println("Donna just bought " + donna.buyTickets(6, booth) +
                           " tickets.");
        System.out.println("Donna is attempting to go on the " + ferris);
        System.out.println("Donna got on: " + donna.getOn(ferris));
        System.out.println("Donna is attempting to go on the " + merryGo);
        System.out.println("Donna got on: " + donna.getOn(merryGo));
        System.out.println("Donna is now a " + donna + "\n");

        System.out.println("Fredy is a " + fredy);
        System.out.println("Fredy just bought " + fredy.buyTickets(5, booth) +
                           " tickets.");
        System.out.println("Fredy is attempting to go on the " + merryGo);
        System.out.println("Fredy got on: " + fredy.getOn(merryGo));
        System.out.println("Fredy is now a " + fredy + "\n");

        System.out.println("Harry is a " + harry);
        System.out.println("Harry just bought " + harry.buyTickets(10, booth) +
                           " tickets.");
        System.out.println("Harry is trying to buy a pass...was he successful: "
                           + harry.buyPass(booth));
        System.out.println("Harry is attempting to go on the " + coaster);
        System.out.println("Harry got on: " + harry.getOn(coaster));
        System.out.println("Harry is attempting to go on the " + tosser);
        System.out.println("Harry got on: " + harry.getOn(tosser));
```

```
        System.out.println("Harry is attempting to go on the " + coaster);
        System.out.println("Harry got on: " + harry.getOn(coaster));
        System.out.println("Harry is now a " + harry + "\n");

        System.out.println("Larry is a " + larry);
        System.out.println("Larry just bought " + larry.buyTickets(15, booth) +
                           " tickets.");
        System.out.println("Larry is attempting to go on the " + tosser);
        System.out.println("Larry got on: " + larry.getOn(tosser));
        System.out.println("Larry is attempting to go on the " + coaster);
        System.out.println("Larry got on: " + larry.getOn(coaster));
        System.out.println("Larry is attempting to go on the " + merryGo);
        System.out.println("Larry got on: " + larry.getOn(merryGo));
        System.out.println("Larry is now a " + larry + "\n");

        System.out.println("Ticket Booth made $" + booth.moneyMade);
        System.out.println(booth);

        System.out.println(coaster + " and had " + coaster.numberOfRiders + " riders.");
        System.out.println(ferris + " and had " + ferris.numberOfRiders + " riders.");
        System.out.println(merryGo + " and had " + merryGo.numberOfRiders + " riders.");
        System.out.println(tosser + " and had " + tosser.numberOfRiders + " riders.");
    }
}
```

Here is the output to expect (pay attention to the bold values to make sure that your code is correct):

```
Ticket booth with 4 passes and 100 tickets

Billy is a 4.9' person with $10.00 and 0 tickets
Billy just bought 20 tickets.
Billy attempting to go on the Roller Coaster requiring 6 tickets with a height restriction of 4.25'
Billy got on: true
Billy attempting to go on the Tummy Tosser requiring 7 tickets with a height restriction of 4.9'
Billy got on: true
Billy is now a 4.9' person with $0.00 and 7 tickets

Donna is a 3.0' person with $5.00 and 0 tickets
Donna is trying to buy a pass...was she successful: false
Donna just bought 6 tickets.
Donna is attempting to go on the Ferris Wheel requiring 5 tickets with a height restriction of 3.1'
Donna got on: false
Donna is attempting to go on the Merry-Go-Round requiring 2 tickets with a height restriction of
0.0'
Donna got on: true
Donna is now a 3.0' person with $2.00 and 4 tickets

Fredy is a 6.0' person with $0.00 and 0 tickets
Fredy just bought 0 tickets.
Fredy is attempting to go on the Merry-Go-Round requiring 2 tickets with a height restriction of
0.0'
Fredy got on: false
Fredy is now a 6.0' person with $0.00 and 0 tickets

Harry is a 4.8' person with $78.50 and 0 tickets
Harry just bought 10 tickets.
Harry is trying to buy a pass...was he successful: true
Harry is attempting to go on the Roller Coaster requiring 6 tickets with a height restriction of
4.25'
Harry got on: true
Harry is attempting to go on the Tummy Tosser requiring 7 tickets with a height restriction of 4.9'
Harry got on: false
Harry is attempting to go on the Roller Coaster requiring 6 tickets with a height restriction of
4.25'
Harry got on: true
Harry is now a 4.8' person with $57.00 and a pass
```

```
Larry is a 4.0' person with $50.00 and 0 tickets
Larry just bought 15 tickets.
Larry is attempting to go on the Tummy Tosser requiring 7 tickets with a height restriction of 4.9'
Larry got on: false
Larry is attempting to go on the Roller Coaster requiring 6 tickets with a height restriction of
4.25'
Larry got on: false
Larry is attempting to go on the Merry-Go-Round requiring 2 tickets with a height restriction of
0.0'
Larry got on: true
Larry is now a 4.0' person with $42.50 and 13 tickets

Ticket Booth made $42.0
Ticket booth with 3 passes and 49 tickets
Roller Coaster requiring 6 tickets with a height restriction of 4.25' and had  3 riders.
Ferris Wheel requiring 5 tickets with a height restriction of 3.1' and had  0 riders.
Merry-Go-Round requiring 2 tickets with a height restriction of 0.0' and had  2 riders.
Tummy Tosser requiring 7 tickets with a height restriction of 4.9' and had  1 riders.
```

NOTE: Submit all **.java** and **.class** files needed to run. You **MUST NOT use packages** in your code, **nor projects**. Submit ALL of your files in one folder such that they can be opened and compiled individually in JCreator. Some IDEs may create packages and/or projects automatically. You **MUST** export the .java files and remove the package code at the top if it is there. Do NOT submit JCreator projects either. **JUST SUBMIT the JAVA and CLASS FILES**. Note that if your internet connection at home is down or does not work, we will not accept this as a reason for handing in an assignment late ... so make sure to submit the assignment WELL BEFORE it is due !

Please NOTE that you WILL lose marks on this assignment if any of your files are missing. You will also lose marks if your code is not written neatly with proper indentation. See examples in the notes for proper style.
