Ticket Class Implementation with IntelliJ

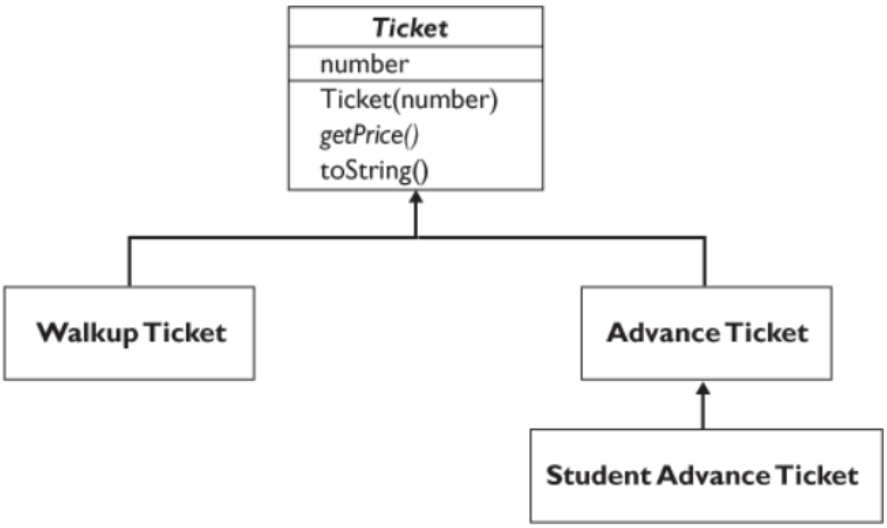
In this exercise you will use pair programming in IntelliJ to implement a ticketing application. Pair programming means that two people work together to complete a programming task. Only one person programs at any given time while the other person observes and provides input/feedback. For this exercise, one person will complete the programming for question 1, 4, 5, and 8 while the other person observes and provides input/feedback. Roles should be swapped for questions 2, 3, 6, and 7. This means that the two programmers will frequently switch roles.

Make sure you use proper access modifiers for your methods and fields. After you finish, submit all java files to BB along with information that clearly identifies your partner for this assignment.

Consider the task of representing types of tickets to campus events. Each ticket has a unique number (the ticket ID) and a price. There are three types of tickets: a walk-up ticket, an advance ticket, and a student advance ticket. You will be implementing each ticket as a class. Each class should be contained in its own file according to the Java convention (i.e., file name matches the class name). Be careful to avoid hard coding values that vary based on ticket type (i.e., the original cost and discount rate).

- The default price for a ticket is $50.
- Walk-up tickets are purchased the day of the event at the default price.
- Advance tickets purchased ten or more days prior to the event are discounted 40% (which would be $30 given a default price of $50). Advance tickets purchased between one and nine days prior to the event are discounted 20% (would would be $40 given a default price of $50)
- Student advance tickets are discounted an additional 50% (i.e., half the price of a normal advance ticket). So, student advance tickets purchased ten or more days prior to an event would cost $15 given a default price of $50, and student advance tickets purchased between one and nine days prior to an event would cost $20 given a default price of $50.

The figure below illustrates the various ticket types (You may add more fields and helper methods as needed – but only do so if they are necessary):



1. Implement a class called Ticket that will serve as the superclass for all three types of tickets. Define all common operations in this class. Define the following operations:
   - The ability to construct a ticket using a unique ticket number. The default price is $50.
   - The ability to construct a ticket using a unique ticket number AND the price of the ticket.
   - The ability to ask for a ticket's price.
   - The ability to println a ticket object as a String. An example String would be "Number: 17, Price: 50.0".

2. Implement an executable class called TicketClient (i.e., having a main method) in a manner that **thoroughly** tests the Ticket class (i.e., instantiate multiple objects of the Ticket class and call various instance methods to confirm functionality).

3. Implement a class called WalkupTicket to represent a ticket purchased the day of an event. Walk-up tickets are also constructed using a unique ticket number. They have the default price. When a walkup ticket is printed, the output should specify the ticket type (for example, "Ticket Type: Walk-up, Number: 17, Price: 15.0 ").

4. Add code to the TicketClient class to **thoroughly** test the WalkupTicket class (i.e., instantiate multiple objects of the WalkupTicket class and call various instance methods to confirm functionality).

5. Implement a class called AdvanceTicket to represent a ticket purchased prior to the day of the event. An advance ticket is constructed using a unique ticket number and the number of days in advance that the ticket was purchased. Advance tickets purchased 10 or more days before the event have 40% discount of the original cost (i.e., $30 given a $50 default price), and advance tickets purchased fewer than 10 days before the event have 20% discount (i.e., $40 given a default price of $50). When an advance ticket is printed, the output should specify the ticket type (for example, "Ticket Type: Advanced, Number: 17, Price: 15.0").

6. Add code to the TicketClient class to **throughly** test the AdvanceTicket class (i.e., instantiate multiple objects of the AdvanceTicket class and call various instance methods to confirm functionality).

7. Implement a class called StudentAdvanceTicket to represent a ticket purchased prior to the day of an event by a student. A student advance ticket is constructed using a unique ticket number and the number of days in advance that the ticket was purchased. Student advance tickets have an additional 50% discount over standard advanced tickets. (So, student advance tickets purchased 10 or more days before the event cost $15, and student advance tickets purchased fewer than 10 days before the event cost $20). When a student advance ticket is printed, the output should mention that the student must show his or her student ID (for example, "Ticket Type: Student Advanced, Number: 17, Price: 15.0 (ID required)").

8. Add code to the TicketClient class to **throughly** test the StudentAdvanceTicket class (i.e., instantiate multiple objects of StudentAdvanceTicket class and call various instance methods to confirm functionality).