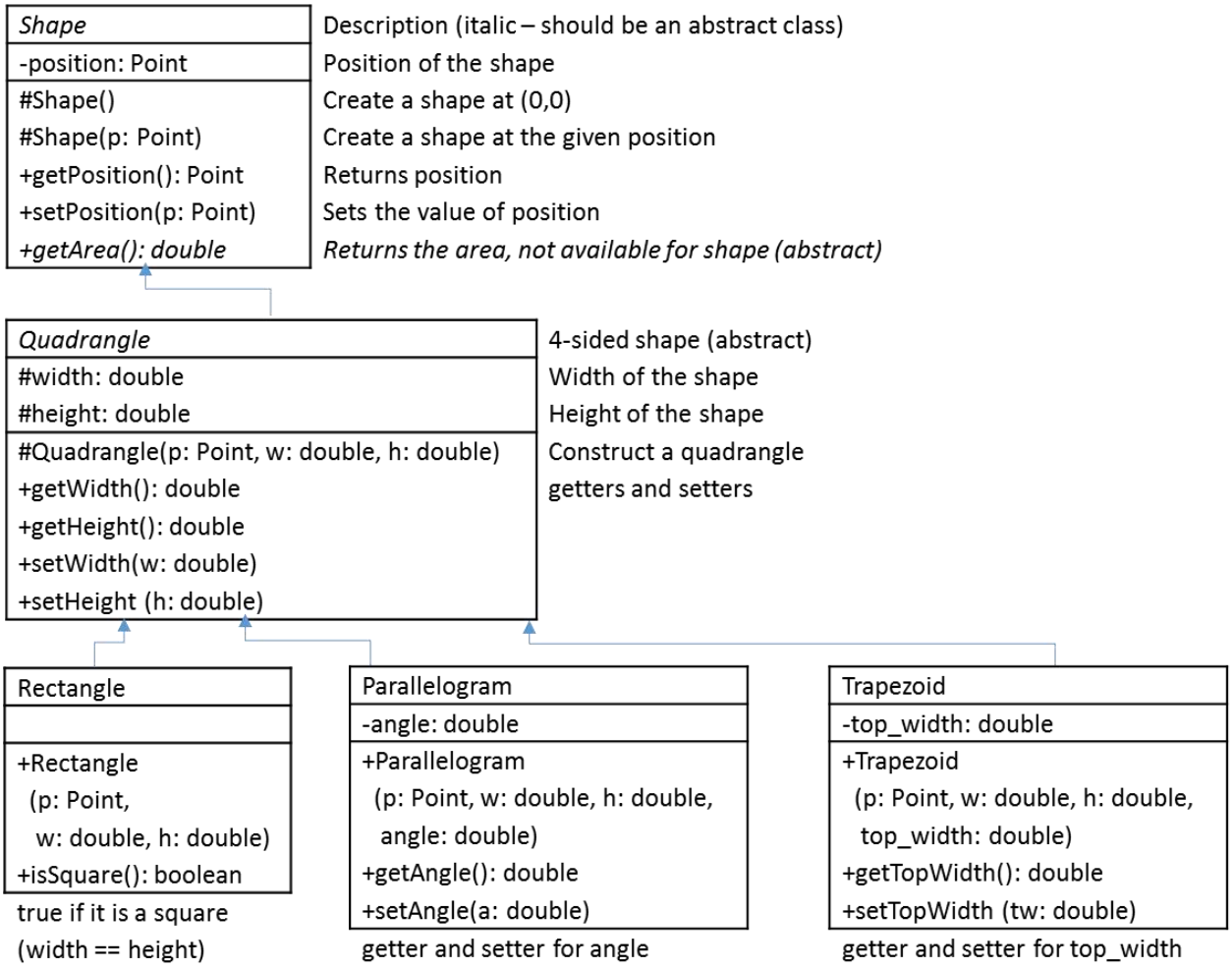
COMP217 Java Programming, spring 2019. Instructor: Gil-Jin Jang

Exercise 12_1 (submission: TestShape.java, 30%)

Design classes `Rectangle`, `Parallelogram`, and `Trapezoid` inherited from an abstract class `Quadrangle`, which is also inherited from class `Shape`.
The abstract method `getArea()` should be defined in all the three classes.
See the following UML diagrams and complete the given code template.

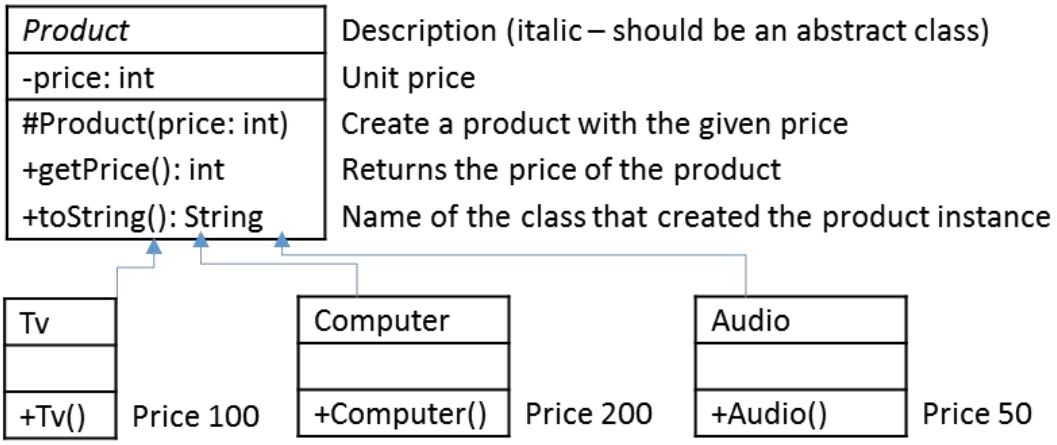


(code template is in the next page)

```
public class TestShape {   /* testing class */
  public static void main(String args[]) {
    Point p = new Point(1,1);
    Shape[] arr = {
      new Rectangle(p,3,4),
      new Parallelogram(p,5,6,Math.PI/6.0),
      new Trapezoid(p,5,6,2)
    };
    System.out.println("SUM_AREA = " + sumArea(arr));
   }

  static double sumArea(Shape[] arr) {
/* FILL IN - sum up the areas of all the shapes using getArea() methods */
   }
}

/* execution results
$ java TestShape
SUM_AREA = 63.0
*/

class Point {       // already completed
  double x, y;
  Point() { this(0,0); }
  Point(double x, double y) { this.x = x; this.y = y; }
  public String toString() { return "["+x+","+y+"]"; }
}

abstract class Shape {
  /* FILL IN */
}

abstract class Quadrangle extends Shape {
  protected double width, height;
  /* FILL IN THE REMAINING PARTS */
}

class Rectangle extends Quadrangle {
  /* FILL IN THE REMAINING PARTS */
  /* The method getArea() should be defined */
}

class Parallelogram extends Quadrangle {
  /* FILL IN THE REMAINING PARTS */
  /* The method getArea() should be defined */
}

class Trapezoid extends Quadrangle {
  /* FILL IN THE REMAINING PARTS */
  /* The method getArea() should be defined */
}
```

Exercise 12_2 (submission: TestBuyer.java, 40%)

Design a buyer class that describes a shopping activity on a limited budget. The necessary classes are given by the following UML expressions.



Buyer	Description
-money: int	The amount of remaining money, initially 0
-cart: Product[]	Bought items, initially null
-nitems: int	Number of items
+Buyer(money: int)	Create a Buyer class with the given money
+buy(p: Product): void	buy an item with checking the remaining money
-add(p: Product): void	add an item to the cart
+print_summary(): void	print current Product items and remaining money

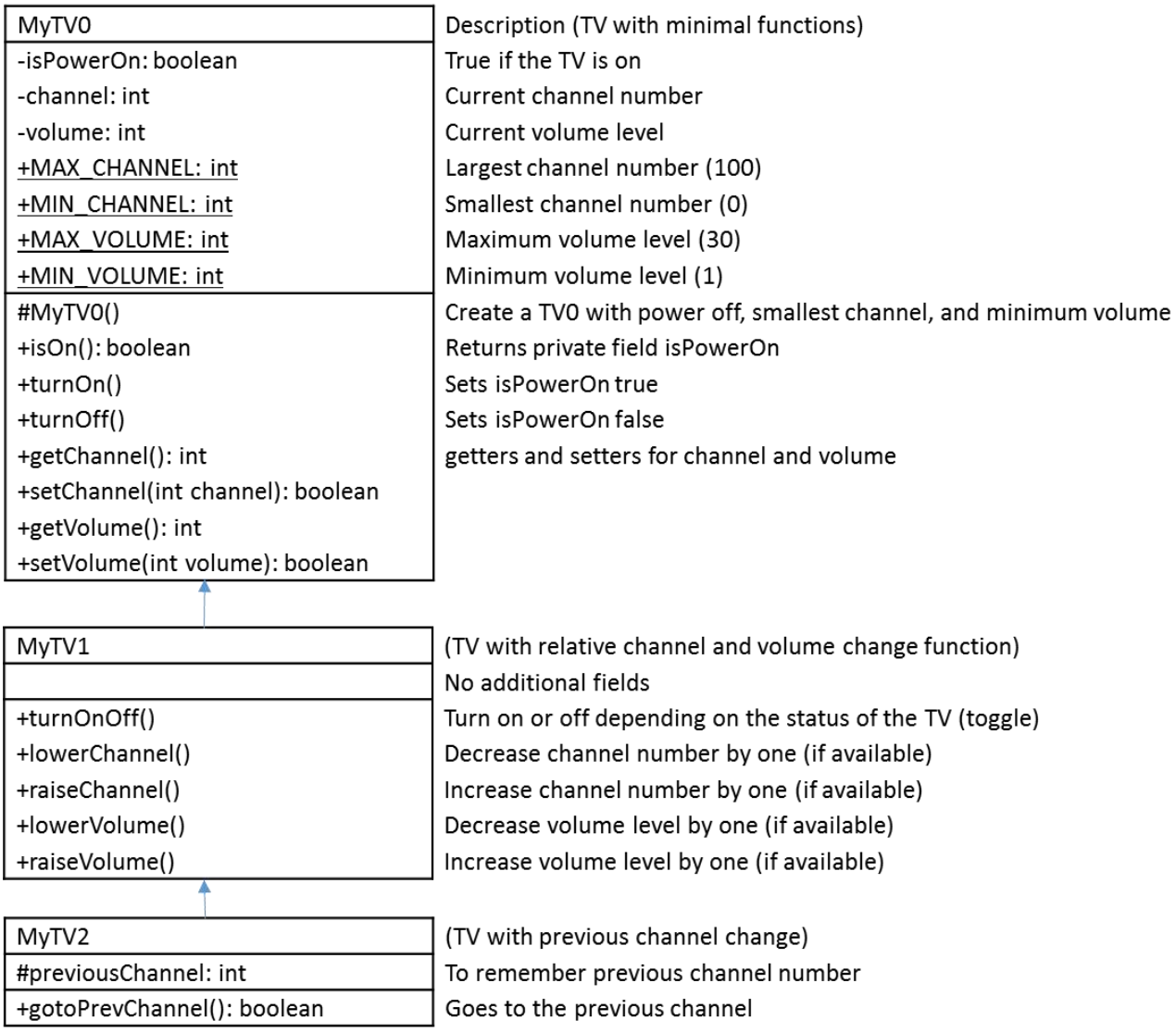
Some of the code segments are given below, and fill in the missing parts to complete the program.

```
public class TestProduct { /* testing class */
  public static void main(String args[]) {
    Buyer b = new Buyer(1000);
    b.buy(new Tv());
    b.buy(new Computer());
    b.buy(new Tv());
    b.buy(new Audio());
    b.buy(new Computer());
    b.buy(new Computer());
    b.buy(new Computer());
    b.print_summary();
  }
}
```

```
/* execution results
$ java TestProduct
NOT_ENOUGH_MONEY_50
Products: Tv Computer Tv Audio Computer Computer
Used money: 850
Remaining money: 150
*/

abstract class Product {   /* already completed */
  private int price;
  protected Product(int price) { this.price = price; }
  public int getPrice() { return price; }
   public String toString() { return this.getClass().getName(); }
       // displays the name of class
}

class Tv extends Product { /* completed */
  public Tv() { super(100); }
}

class Computer extends Product {
  /* FILL IN - the price of a computer is 200 */
}

class Audio extends Product {
  /* FILL IN - the price of an audio is 50 */
}

class Buyer {
  /* FILL IN
  Requirements:
  -  All the field variables are private
  -  Method buy(Product p): buys a product given as an argument. Subtract
     the product price from money, and add it to cart by calling method
     add(p). If the money is less than the price, do not either add nor
     subtract money.
  -  Method add(Product p): if the array cart is null, create cart with a
     single item (new Product[1]). If it is full, increase the array size
     two times (e.g. 1->2, 2->4, etc.). You should keep the old products as
     well.
  -  'nitems' is the actual number of products in the cart, which is less
     than or equal to cart.length
  -  Method print_summary(): displays bought items, Used money, and
     Remaning money
  -  Try not to add any extra fields or methods.
  Input validation:
  -  Display 'NOT_ENOUGH_MONEY_(required amount)' when the remaining money
     is not enough
  */
}
```

Exercise 12_3 (submission: TestMyTV0.java, 30%)

Complete designing classes `MyTV0`, `MyTV1`, and `MyTV2`, according to the following UML diagrams. Complete the given code template.



```
public class TestMyTV0 {   /* testing class */
  static void print_tv(MyTV0 t) {
    if (t.isOn())
      System.out.println("CH:"+t.getChannel()+" VOL:"+t.getVolume());
    else  System.out.println("TV_OFF");
  }
  public static void main(String args[]) {
    MyTV2 t = new MyTV2(); // initially, MIN_CHANNEL 1 MIN_VOLUME 0

    t.setChannel(100);      // OFF, ignored
    t.setVolume(0);  // OFF, ignored
    print_tv(t);     // TV_OFF

    t.turnOnOff();   // ON default 1 1
    t.raiseChannel();       // 2 1
    t.raiseVolume(); // 2 2
    print_tv(t);     // CH 2 VOL 2

    t.setChannel(10);       // 10 2
    t.setVolume(100);       // 10 100
    t.lowerChannel();       // 9 100
    t.lowerVolume(); // 9 99
    print_tv(t);     // CH 9 VOL 99

    t.turnOnOff();   // OFF
    t.lowerChannel();       // OFF, ignored
    t.lowerVolume(); // OFF, ignored
    print_tv(t);     // TV_OFF

    t.turnOnOff();   // ON 9 99
    t.setChannel(50);       // 50 99
    print_tv(t);     // CH 50 VOL 99
    t.gotoPrevChannel();    // 9 99
    print_tv(t);     // CH 9 VOL 99
    t.gotoPrevChannel();    // 50 99
    print_tv(t);     // CH 50 VOL 99
    t.lowerChannel();       // 49 99
    t.lowerChannel();       // 48 99
    print_tv(t);     // CH 48 VOL 99
    t.gotoPrevChannel();    // 49 99
    print_tv(t);     // CH 49 VOL 99
    t.gotoPrevChannel();    // 48 99
    print_tv(t);     // CH 48 VOL 99
  }
}
```

```
/* execution output - updated on 5/28, 2018
$ java hw3_3
TV_OFF
CH:2 VOL:2
CH:9 VOL:99
TV_OFF
CH:50 VOL:99
CH:9 VOL:99
CH:50 VOL:99
CH:48 VOL:99
CH:49 VOL:99
CH:48 VOL:99
*/



class MyTV0 {       /* all the fields are already given */
  private boolean isPowerOn;
  private int channel;
  private int volume;

  /* error fixed on 5/16/2018 12:23pm, default changed on 5/25/2018 */
  public static final int MAX_CHANNEL = 100;
  public static final int MIN_CHANNEL = 1;
  public static final int MAX_VOLUME = 100;
  public static final int MIN_VOLUME = 1;

  /* FILL IN - implement
     * 1. constructor: MyTV0()
     * 2. methods for power management: isOn(), turnOn(), turnoff()
     * 3. getters and setters for channel and volume
     *  - The default values for the constructor are:
     *      isPowerOn = false,
     *      channel = MIN_CHANNEL,
     *      volume = MIN_VOLUME
     *  - In setChannel(int channel),
     *    (a) the TV should be on to set the channel number
     *    (b) the channel should be between MIN_CHANNEL and MAX_CHANNEL
     *  - In setVolume(int volume),
     *    (a) the TV should be on,
     *    (b) the volume should be between MIN_VOLUME and MAX_VOLUME
  */

  ///////// START OF FILE-IN ////////////

    // ADD YOUR CODE

    ///////// END OF FILE-IN ////////////
}
```

```
class MyTV1 extends MyTV0 {
  /*   /* ALREADY IMPLEMENTED - NO NEED TO FILE-IN
     * 1. turnOnOff(): reverses the value of isPowerOn.
     *    However, because isPowerOn is private field of MyTV0,
     *    getter and setter functions should be used.
     * 2. lowerChannel(), raiseChannel(): change the channel number by +/-1.
     *    Class variable channel is private, so cannot be changed directly.
     * 3. lowerVolume(), raiseVolume(): change the volume level by +/-1.
     *    Class variable volume is private, so cannot be changed directly.
     * */


   public void turnOnOff() { if ( isOn() ) turnOff(); else turnOn(); }

  public void lowerChannel() { setChannel(getChannel()-1); }
  public void raiseChannel() { setChannel(getChannel()+1); }
  public void lowerVolume() { setVolume(getVolume()-1); }
  public void raiseVolume() { setVolume(getVolume()+1); }
}

class MyTV2 extends MyTV1 {
  /* FILL IN
     * 1. New field previousChannel is required
     * 2. gotoPrevChannel(): change the channel using the value of
     *    previousChannel and setter function.
     * 3. May have to override the channel setter function
     * */

    ///////// START OF FILE-IN ////////////

    // ADD YOUR CODE

    ///////// END OF FILE-IN ////////////
}
```