AP Computer Science A:
Mutator Methods

CodeHS

Recap: Instance Variables

Access to attributes should be kept internal to the class. Instance variables are designated as **`private`** so as to prevent users from directly manipulating the state of an object.

```
public class Rectangle
{
    private int width;
    private int height;

}
```

Recap: Accessors and Mutators

Rather than allow users to manipulate data directly, we use **accessor** (getter) and **mutator** (setter) methods to control what aspects of an object can be altered:

```
public int getWidth()
{
    return width;
}
```

```
public void setWidth(int newWidth)
{
    width = newWidth;
}
```

Recap: Accessors and Mutators

This lesson will dive deeper into Mutator methods!

```
public void setWidth(int newWidth)
{
    width = newWidth;
}
```

Mutators Methods

Mutator methods are often void methods that change the value of instance and static variables:

```
public void setWidth(int newWidth)
{
    width = newWidth;
}
```

If a method has no return value, the return type is **void**

Mutators Methods

Mutator methods are often void methods that change the value of instance and static variables:

```
public void setWidth(int newWidth)
{
    width = newWidth;
}
```

Mutator methods alter the value of **instance** and **static** variables.

Mutators Methods

Mutator methods don't necessarily need parameters:

```
public void setWidth(int newWidth)
{
    width = newWidth;
}
```

```
public void setWidth()
{
    width++;
}
```

Practice Problem

Let's create Mutator methods for the Power class:

```
public class Power
{
     private String namePower;
     private int strength;

     public Power(String name, int theStrength)
     {
          namePower = name;
          strength = theStrength;
     }
}
```

Practice Problem

Let's create Mutator methods for the `Power` class:

1. setStrength(int newStrength)

2. setName(String name)

Practice Problem

Let's create Mutator methods for the Power class:

```
public void setStrength(int newStrength)
{
    strength = newStrength;
}
```

```
public void setName(String name)
{
    namePower = name;
}
```

Practice Problem

Now let's create mutator methods for the SuperHero class:

```
public class SuperHero
{
     private String name;
     private Power superPower;

     public SuperHero(String heroName, Power power)
     {
          name = heroName;
          superPower = new Power(power.getName(), power.getStrength());
     }

}
```

Practice Problem

Now let's create mutator methods for the SuperHero class:

1. setName(String heroName)
2. setPower(String name, int strength)

Practice Problem

Now let's create mutator methods for the SuperHero class:

```
public void setName(String heroName)
{
    name = heroName;
}
```

```
public void setPower(String name, int strength)
{
    superPower.setName(name);
    superPower.setStrength(strength);
}
```

Practice Problem

Now let's create mutator methods for the SuperHero class:

```
public void setName(String heroName)
{
    name = heroName;
}
```

```
public void setPower(String name, int strength)
{
    superPower.setName(name);
    superPower.setStrength(strength);
}
```

We need to use the setter methods from the Power class to set the name and strength from the SuperHero class!

Now It's Your Turn!

CodeHS

Concepts Learned this Lesson

Term	Definition
Mutator Method	A method that enables user to change the value of an object's instance and static variables.

Standards Covered

- **(LO) MOD-2.E Define behaviors of an object through void methods with or without parameters written in a class.**
- **(EK) MOD-2.E.1 A void method does not return a value. Its header contains the keyword void before the method name.**
- **(EK) MOD-2.E.2 A mutator (modifier) method is often a void method that changes the values of instance variables or static variables.**