

> Summary and Outcomes

Students will learn to create accessor and mutator methods within classes. The accessor methods will return information about an instantiated object of that class. Students will define the type of data to be returned and will designate this as the return type. The mutator methods will send information into an object or change existing instance variables, passing information by means of pre-defined argument types. To accomplish all of this, students will design and implement a `VendingMachine` class.

> Student Assessment

How well do students perform each of the following tasks:

- Complete the `VendingMachine` class to include methods to obtain and set the product price
- Add another property of their choosing and its associated methods
- Comment as needed, ensuring that each method has a description of what it does
- Save, compile, and test often

> Prerequisite Knowledge and Skills

BlueJ:

- Creating projects
- Adding new classes
- Instantiating objects
- Testing methods from instantiated objects

Java:

- Understanding class structure and methods

> This Project Targets the Following Subject Areas(s):

Pre-Java	Java Programming
<input type="checkbox"/> Hardware Basics	<input type="checkbox"/> Applet Programming
<input checked="" type="checkbox"/> Software Basics	<input type="checkbox"/> Subroutine Programming
<input type="checkbox"/> Networks and Servers	<input checked="" type="checkbox"/> Full Scale Programming
<input type="checkbox"/> HTML	
<input type="checkbox"/> Action Scripting	
<input type="checkbox"/> Java Scripting	

> Curriculum-Framing Questions

Essential Question	<ul style="list-style-type: none"> • What does a method return?
Activity Questions	<ul style="list-style-type: none"> • What are the return types? • When are void types used?
Sample Content Questions	<ul style="list-style-type: none"> • What is the syntax of a return type? • Can a method return more than one value?

> Targeted Content Standards, Benchmarks, or State Frameworks

Colorado State Standards:

Mathematics:

- **Standard 2** – Students use algebraic methods to explore, model, and describe patterns and functions involving numbers, shapes, data, and graphs in problem-solving situations and communicate the reasoning used in solving these problems.
 - Grades 9–12 – Analyzing and explaining the behaviors, transformations, and general properties of types of equations

International Society of Technology Education (ISTE):

Information Literacy Standards:

- **Standard 6** – Mathematics instructional programs should focus on solving problems as part of understanding mathematics so that all students
 - Build new mathematical knowledge through their work with problems
 - Develop a disposition to

and functions.

formulate, represent, abstract,
and generalize in situations
within and outside mathematics

- Apply a wide variety of strategies to solve problems and adapt the strategies to new situations
- Monitor and reflect on their mathematical thinking in solving problems

Technology Foundation Standards for Students:

- **Standard 3** – Students use technology tools to enhance learning, increase productivity, and promote creativity. Students use productivity tools to collaborate in constructing technology-enhanced models, prepare publications, and produce other creative works.
- **Standard 6** – Students use technology resources for solving problems and making informed decisions.

> Materials and Resources Required for Project

Equipment	<ul style="list-style-type: none"> • Computer for each student with BlueJ and Java SDK installed
Consumable Supplies	<ul style="list-style-type: none"> • Diskettes • Printed output for prototype
Textbooks/Lesson Guides	<ul style="list-style-type: none"> • <i>Swing, Second Edition</i> by Matthew Robinson, et al. Chapter 5 for treatment of Java / html parameter passing is a great example of using Java to highlight links. • <i>Core Web Programming</i> by Marty Hall and Larry Brown. Chapter 9 for discussion and examples of applets used in web programming
Technology	<ul style="list-style-type: none"> • Windows-based machines running Windows 98 SE or higher

> Activity Outline

Overview: In this unit, students will learn about class methods, with a special focus on return types. They will discover that some methods simply display information while others perform calculations. The former return a “void” type. The latter return results of calculations, with a return type the same as the type of calculation. Students will also be introduced to *accessor* and *mutator* methods (sometimes called getters and setters). Because *accessor* methods get something, their return type is the same as the data the method is retrieving. In contrast, *mutator* methods, which change instance data, do not return anything. As students code the following `VendingMachine` example, they will classify the types of methods, and they will need to understand what the return types will be and why. Finally, students will add new methods and they will need to understand into which categories these methods fall (*accessor*, *mutator*, calculations, display).

Begin by coding and compiling the simple `VendingMachine` example.

```

/*
 * VendingMachine.java
 * This class simulates a vending machine. Each machine can sell
 * one type of product at a fixed price.
 * @author (Your name here)
 * @version (Today's date here)
 */

public class VendingMachine {
    // instance variables
    private String productType; // type of product
    private int productPrice; // product's price
    private int numberSold; // total number of units
                                // sold
    private int balance; // amount of money buyer
                        // inserted
    private int totalMoney; // total $ taken by this
                            // machine

    //constructor
    //the public methods:
    /**
     * Method to set this machine's type of product

```

```

    */
    public void setType(String type) {
        productType = type;
    } // end of setType()

    /**
     * Return this machine's type of product
     */

    public String getType() {
        return productType;
    } // end of getType()

    /**
     * Allow buyer to insert money
     */

    public void insertMoney(int money) {
        balance += money;
    } // end of insertMoney()

    /**
     * give the buyer the product
     */

    public void giveProduct() {
        System.out.println("*-----*");
        System.out.println("| You bought |");
        System.out.println("| |");
        System.out.println("| " + productType);
        System.out.println("|-----|");
        numberSold++; // increase number sold by one
    } // end of giveProduct()

    /**
     * Give the buyer's change back after giving product
     */

    public void giveChange() {
        balance -= productPrice; //how much to give back
        System.out.println("Your change is: " + balance);

        balance = 0; // clear balance
    } // end of giveChange()
} // end of VendingMachine class

```

Once you have entered and compiled the class, create a new instance of the `VendingMachine` class and test it.

Sun, Sun Microsystems, the Sun Logo are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. This Java activity was developed for The Sun Foundation by Boulder High School 2004.

Tips, Hints, and Tricks: Get creative and make the program more robust. Add new methods to make the program more flexible. Consider adding some error checking.

> **Pacing/Timeline**

- Code and debug basic `VendingMachine` class: 1 day
- Design, implement, and debug additional methods: 2–3 days

> **Teacher Reflection** (For example, what worked well in this lesson? What would you change if you were to teach it again?)

- Did students understand the role of methods and return types?
- Were they able to develop methods with arguments passed into objects?
- Did students have additional suggestions for methods?

> Student Handout: Java Programming Java Return Types



- It is often useful for an object to return information about itself. You can do this by creating methods that send back particular information about the object. You need to define the information's data type in advance. This is known as the method's return type. Likewise, you can send information to an object through a method by way of arguments. Each argument has a particular data type defined in advance. This is known as the argument type. You can also invoke methods to display information or perform a calculation.
- The various method and return types are outlined in all of the most popular Java texts. In this unit, you will have the opportunity to see defined methods in action and create additional ones.
- Create a new project called Vending and a `VendingMachine` class. Open the class in the editor and enter the code that follows. Remember to compile your class as you go along.

```
/**
 * VendingMachine.java
 * This class simulates a vending machine. Each machine
 * can sell one type of product at a fixed price.
 * @author (Your name here)
 * @version (Today's date here)
 */
public class VendingMachine
{
    // instance variables
    private String productType; // type of product
    private int productPrice; // product's price
    private int numberSold; //total number of units sold
    private int balance; //amount of money buyer inserted
    private int totalMoney; //total $ taken by this machine

    //constructor
    public VendingMachine()
    {
        balance = 0;
        totalMoney = 0;
        numberSold = 0;
    }
}
```

Sun, Sun Microsystems, the Sun Logo are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. This Java activity was developed for The Sun Foundation by Boulder High School 2004.

```

        } // end of constructor
//the public method
* Method to set this machine's type of product
*/
public void setPrice(int price)
{
    productPrice = price;
} // end of setPrice()
//the public method
* Method to set this machine's type of product
*/
public void setType(String type)
{
    productType = type;
} // end of setType()
/**
 * Return this machine's type of product
 */
public String getType()
{
    return productType;
} // end of getType()
/**
 * Allow buyer to insert money
 */
public void insertMoney(int money)
{
    balance += money;
} // end of insertMoney()
/**
 * give the buyer the product
 */
public void giveProduct()
{
    System.out.println("*-----*");
    System.out.println("| You bought |");
    System.out.println("| |");
    System.out.println("| " + productType);
    System.out.println("|-----|");
    numberSold++; // increase number sold by one
} // end of giveProduct()
/**
 * Give the buyer's change back after giving product
 */
public void giveChange()
{
    balance -= productPrice; //how much to give back
    System.out.println("Your change is: " + balance);
    balance = 0; // clear balance
}

```

```
} // end of giveChange()  
} // end of VendingMachine class
```

- Create a new instance of the `VendingMachine` class and test all of the methods.

> YOUR JOB:

- Complete the `VendingMachine` class to include methods to get and set the product price
- Add another property of their choosing and the associated methods
- Comment where appropriate
- Save, compile, and test OFTEN!



> EXTRAS YOU CAN ADD:

- Get creative and make the program better. Add more properties and methods to make the machine more flexible.