Unit 7 - Lesson 1
Project Planning

Computer Science A

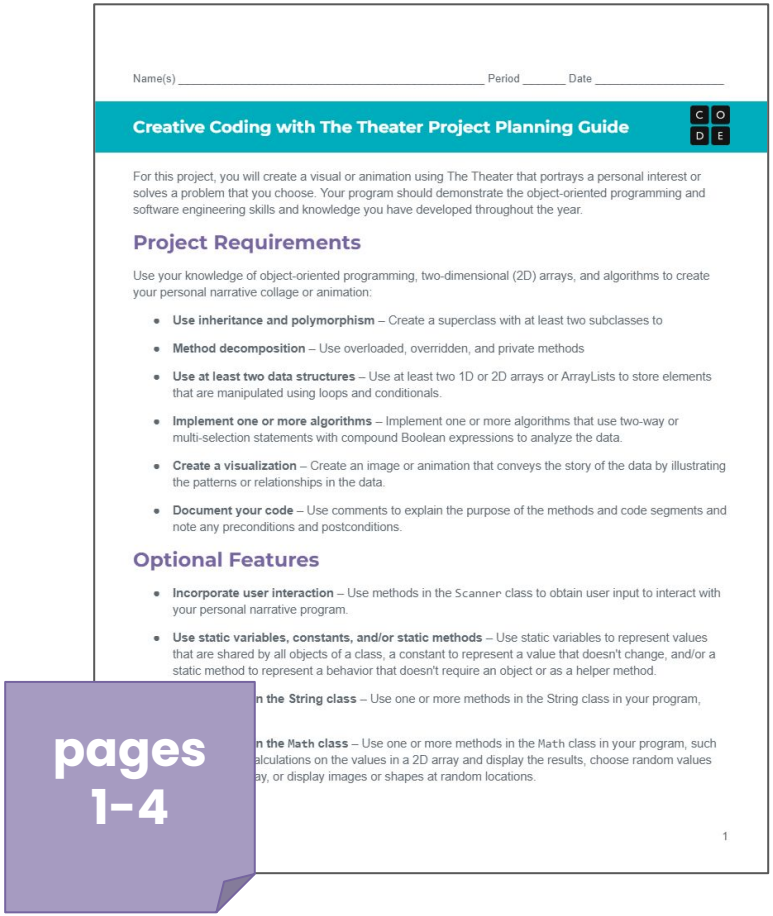
🚀 Question of the Day

How can I keep track of the tasks I need to complete for a project?

The Creative Coding with The Theater Project is an open-ended project that allows you to choose the topic for your project and the type of visual or animation you want to create.

✅ Do This:

Read through the **Project Description** on the first page of the **Creative Coding with The Theater Project Planning Guide** and the **rubric** on pages 3-4.



Name(s) _____ Period _____ Date _____

Creative Coding with The Theater Project Planning Guide

For this project, you will create a visual or animation using The Theater that portrays a personal interest or solves a problem that you choose. Your program should demonstrate the object-oriented programming and software engineering skills and knowledge you have developed throughout the year.

Project Requirements

Use your knowledge of object-oriented programming, two-dimensional (2D) arrays, and algorithms to create your personal narrative collage or animation:

- **Use inheritance and polymorphism** – Create a superclass with at least two subclasses to
- **Method decomposition** – Use overloaded, overridden, and private methods
- **Use at least two data structures** – Use at least two 1D or 2D arrays or ArrayLists to store elements that are manipulated using loops and conditionals.
- **Implement one or more algorithms** – Implement one or more algorithms that use two-way or multi-selection statements with compound Boolean expressions to analyze the data.
- **Create a visualization** – Create an image or animation that conveys the story of the data by illustrating the patterns or relationships in the data.
- **Document your code** – Use comments to explain the purpose of the methods and code segments and note any preconditions and postconditions.

Optional Features

- **Incorporate user interaction** – Use methods in the `Scanner` class to obtain user input to interact with your personal narrative program.
- **Use static variables, constants, and/or static methods** – Use static variables to represent values that are shared by all objects of a class, a constant to represent a value that doesn't change, and/or a static method to represent a behavior that doesn't require an object or as a helper method.
- **n the String class** – Use one or more methods in the String class in your program,
- **n the Math class** – Use one or more methods in the `Math` class in your program, such alculations on the values in a 2D array and display the results, choose random values ay, or display images or shapes at random locations.

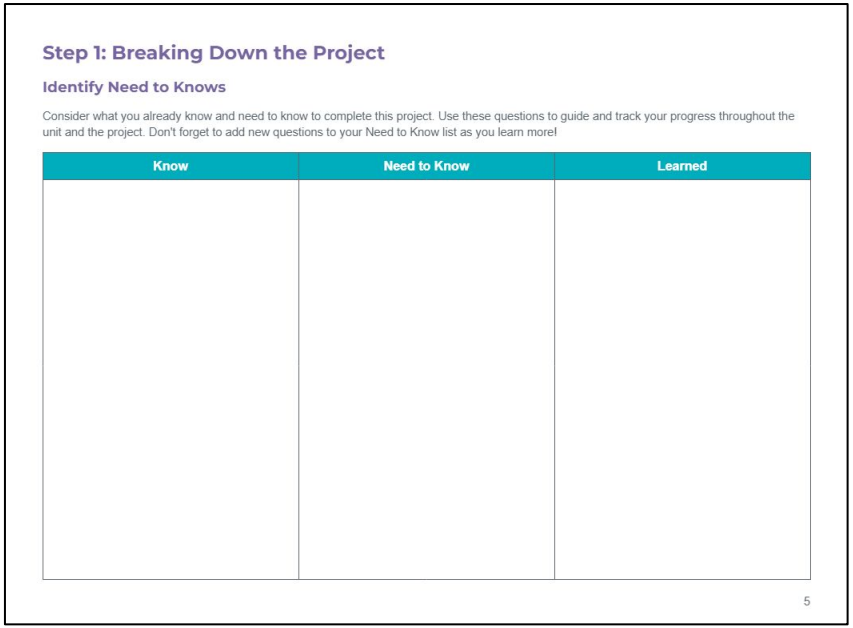
1

pages 1-4

✅ Do This:

Go to **page 5** of the **Project Planning Guide**.

Fill in the **Know** column with what you already know about Java that you can use to complete this project.



Step 1: Breaking Down the Project

Identify Need to Knows

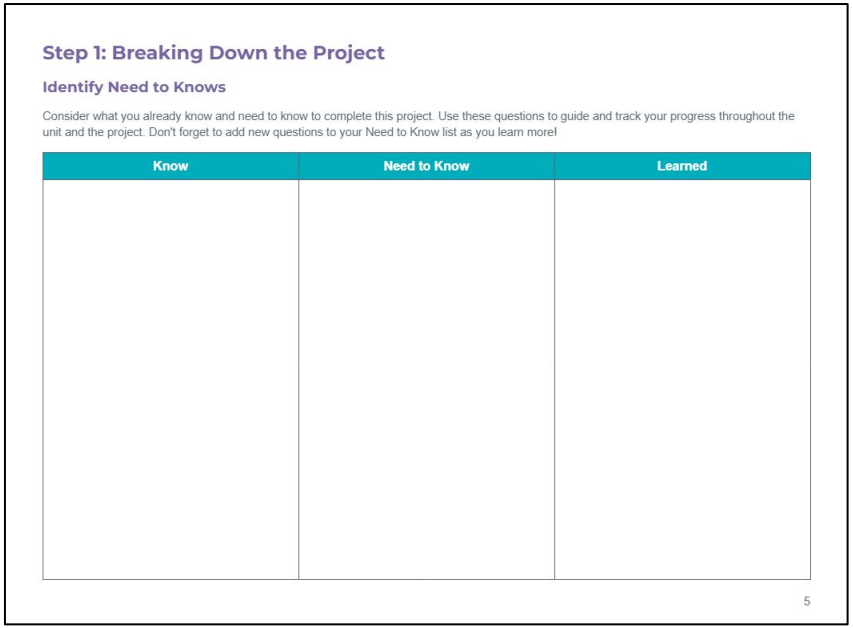
Consider what you already know and need to know to complete this project. Use these questions to guide and track your progress throughout the unit and the project. Don't forget to add new questions to your Need to Know list as you learn more!

Know	Need to Know	Learned

5

✅ Do This:

Fill in the **Need to Know** column with questions you have or things you don't know that you will need to learn to complete this project.



Step 1: Breaking Down the Project

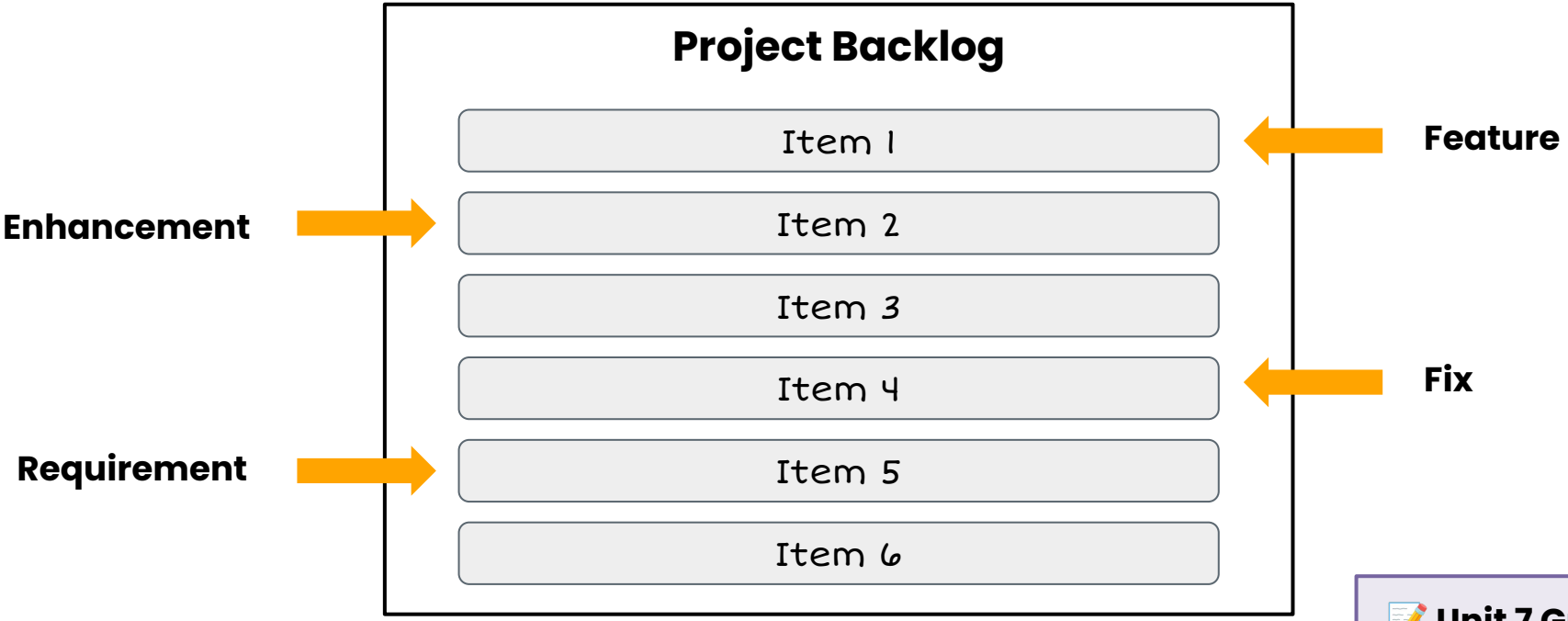
Identify Need to Knows

Consider what you already know and need to know to complete this project. Use these questions to guide and track your progress throughout the unit and the project. Don't forget to add new questions to your Need to Know list as you learn more!

Know	Need to Know	Learned

5

In software development, a **project backlog** is a prioritized list of tasks to complete for a project.



📝 Unit 7 Guide

A **benchmark** is a standard or point of reference to assess progress.

📝 Unit 7 Guide

🎯 Benchmark #1: Due Lesson 6

- Brainstorm project ideas and goals
- Decompose the problem to identify the classes and methods you will need to implement
- Obtain and implement feedback from peers

🎯 Benchmark #2: Due Lesson 8

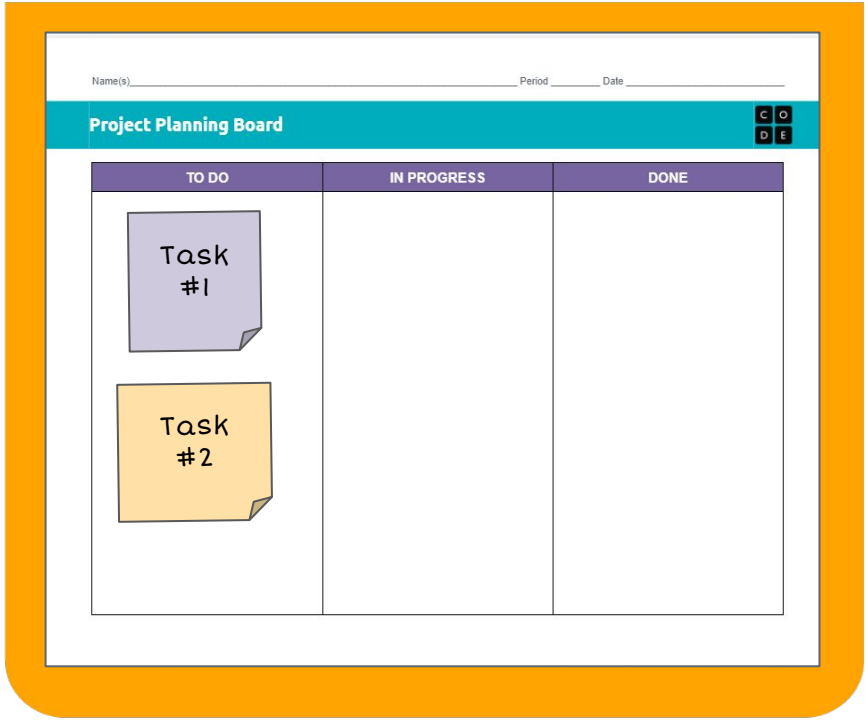
- Develop one or more classes and data structures
- Implement one or more algorithms to create a visual or animation
- Obtain and implement feedback from peers

🎯 Benchmark #3: Due End of Project

- Self-assess your work using the program requirements and rubric
- Finalize your program using the feedback you received and your self-assessment
- Showcase your work!

✅ Do This:

Complete the first benchmark in the **TO DO** column of your Project Planning Board.



💼 Key Vocabulary

- **benchmark:** a standard or point of reference to assess progress
- **project backlog:** a prioritized list of tasks to complete for a project

Unit 7 - Lesson 2
Object References as Parameters

Computer Science A

🚀 Question of the Day

What are the similarities and differences between passing primitives and object references as parameters?

```
public Painter(int x, int y, String dir, int paint)
```

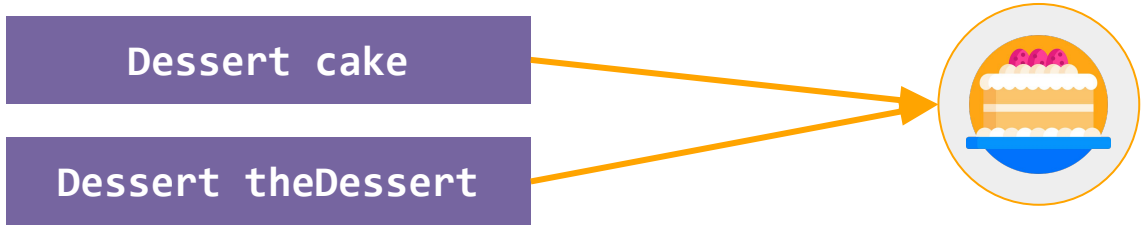
A **formal parameter** is the **value to be passed** to a constructor or method.

```
Painter katie = new Painter(2, 3, "North", 4);
```

An **actual parameter** is the **value to assign** to the formal parameter.

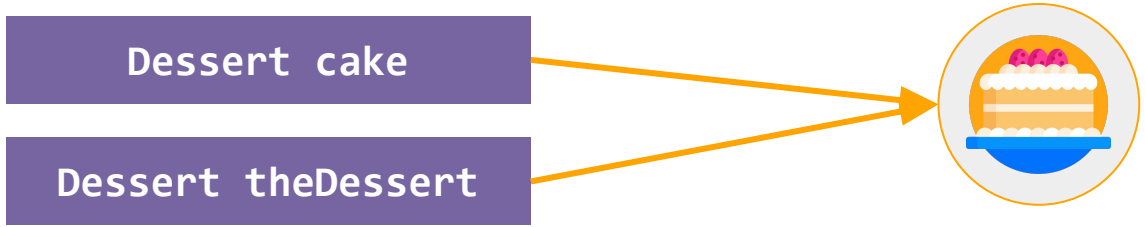
When the **actual parameter** is an **object reference**, a **copy** of the **reference** is passed.

```
joy.applyDiscount(cake);
public void applyDiscount(Dessert theDessert) { . . . }
```



The **actual** and **formal parameters** are then **aliases**.

```
joy.applyDiscount(cake);
public void applyDiscount(Dessert theDessert) { . . . }
```



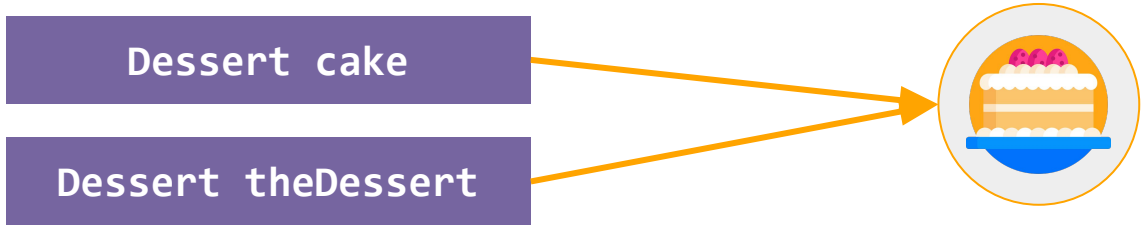
When the **actual parameter** is a **primitive**, a **copy** of the **value** is passed.

```
cake.setPrice(2.99);
public void setPrice(double newPrice) { . . . }
```

	2.99
newPrice	2.99

This process is called **pass by value** , which means that a **copy** of the **actual value** is passed to a constructor or method.

```
joy.applyDiscount(cake);
public void applyDiscount(Dessert theDessert) { . . . }
```



BankAccount.java

```
public class BankAccount {
  private double balance;

  private void transfer(BankAccount other, double amt) {
    if (balance >= amt) {
      balance -= amt;
      other.balance += amt;
    }
    else {
      System.out.println("Insufficient funds.");
    }
  }
}
```

Methods can only access the **private** instance variables and methods of the **parameter** when it is the **same type** as the class it is in.

```
BankAccount checking = new BankAccount(100);
BankAccount savings = new BankAccount(50);
checking.transfer(savings, 50);
```

HOLD that THOUGHT

Discuss:

What does the **`this`** keyword do when it is used as a parameter?

Student.java

```
public class Student {
  . . .
  public void introduce() {
    System.out.println("Hi, my name is " + name);
    describeHobby(this);
  }

  public void describeHobby(Student student) {
    System.out.println("I like to " + student.hobby);

  }
}
```

When the **this** keyword is used as a parameter, the current object (or **this** object) is passed.

```
Student monique = new Student("Monique", "paint");
monique.introduce();
```

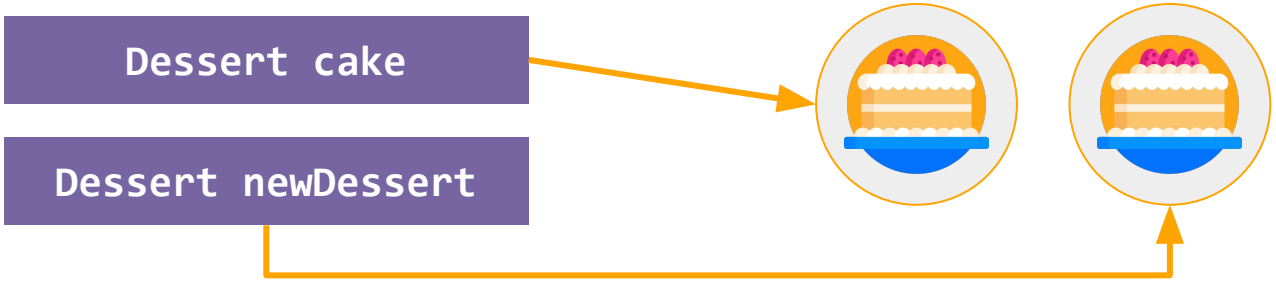
💡 Discuss:

When should we **avoid** passing objects as parameters? Why?

```
Dessert cake = new Dessert();
FoodTruck joysTruck = new FoodTruck(cake);
```

```
public FoodTruck(Dessert newDessert) {
  this.newDessert = new Dessert(newDessert.getFlavor(), newDessert.getPrice();
}
```

When an **object reference** is a **constructor parameter**, the **instance variable** should be initialized with a **copy** of the **referenced object**.



An **end user** is the person who will use the program.

A **user story** is an informal explanation of a program feature written from the perspective of the user.

🎥 Software Engineering: User Stories

Why are user stories important in software design and development?

Complete the guided notes on the 📝 **Unit 7 Guide**.

Example User Stories

As a food truck owner, I want to enter dessert inventory, so I know when certain items are low.

As a student, I want to organize my homework, so I can manage my deadlines and projects.

As a content creator, I want to know which posts my followers like best, so I can create more posts like those.

💡 Discuss:

Think about an app or program you use every day. Who are the **end users** for that app or program?

Retrieve
your knowledge and ideas and write it down silently

Pair
up with a neighbor and talk about your reflections

Share
your thoughts in a class discussion

💡 Discuss:

What might a **user story** be for that app or program?

✅ Do This:

Write **user stories** for your project on page 8 of your Project Planning Guide.

page 8

Step 3: User Stories

A **user story** is a short explanation of a program feature that is written from the end user's point of view to describe how the feature will provide value to the user. User stories are often expressed in short sentences, like **"As a [persona], I [want to], [so that]."** You should write a user story for each large step or component of your overall project. For each user story:

- Identify the criteria for what it means for it to be considered "done".
- Decide which specific steps need to be completed for it to be considered "done".

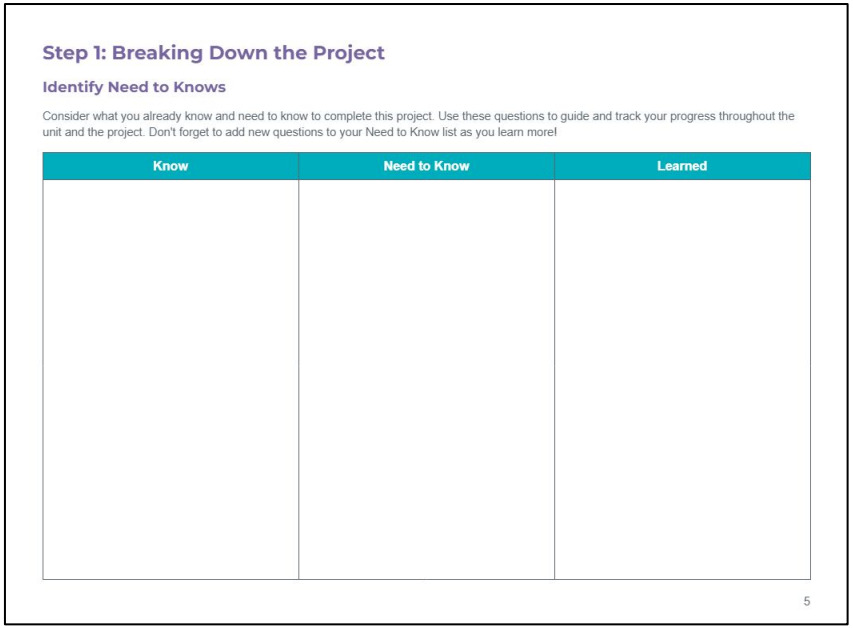
User Story	Criteria	Steps to Complete

8

✅ Do This:

Revisit your **Need to Knows**!

- Check off **answered questions** in the **Need to Know** column.
- Add what you have **learned** and **answers to any questions** in the **Learned** column
- Add any **new questions** to the **Need to Know** column



Step 1: Breaking Down the Project

Identify Need to Knows

Consider what you already know and need to know to complete this project. Use these questions to guide and track your progress throughout the unit and the project. Don't forget to add new questions to your Need to Know list as you learn more!

Know	Need to Know	Learned

5

💼 Key Vocabulary

- **end user:** the person who will use the program
- **pass by value:** the process of making a copy of the actual value of a variable to pass to a constructor or method
- **user story:** an informal explanation of a program feature written from the perspective of the user

Unit 7 - Lesson 3
Overloaded Methods

Computer Science A

Warm Up

💡 **Discuss:** How can we retrieve a club member's **first** and **last name** from their **username**?

For example, Grace Hopper's username would be **`Grace.Hopper`**.

`String` Methods

```
indexOf(String str)
substring(int beginIndex)
substring(int beginIndex, int endIndex)
```

📝 **Unit 7 Guide**

```
// username = "Grace.Hopper";

int index = username.indexOf(".");
// index = 5

String first = username.substring(0, index);
// first = "Grace.Hopper".substring(0, 5);
// first = "Grace";

String last = username.substring(index + 1);
// last = "Grace.Hopper".substring(6);
// last = "Hopper";
```

💡 Discuss:

- What is the difference between the **two versions** of `substring()`?
- How does Java know **which one to use**?

💡 Discuss:

Why do you think the designers of Java decided to give the **two versions** the **same name**?

🚀 Question of the Day

How can overloading a method be useful in my programs?

```
public class Dessert {

  public Dessert() {
      flavor = "glazed";
      price = 1.50;
  }

  public Dessert(String flavor, double price) {
     this.flavor = flavor;
     this.price = price;
  }

}
```

Dessert()

Dessert("Strawberry", 3.00)

Overloading: defining two or more constructors or methods with the same name but different signatures

Signature: consists of the name and parameter list

💡 Discuss:

- Why did we use overloading with constructors?
- How might overloading be helpful with methods?

Methods are said to be **overloaded** when there are multiple methods with the same name but a different signature.

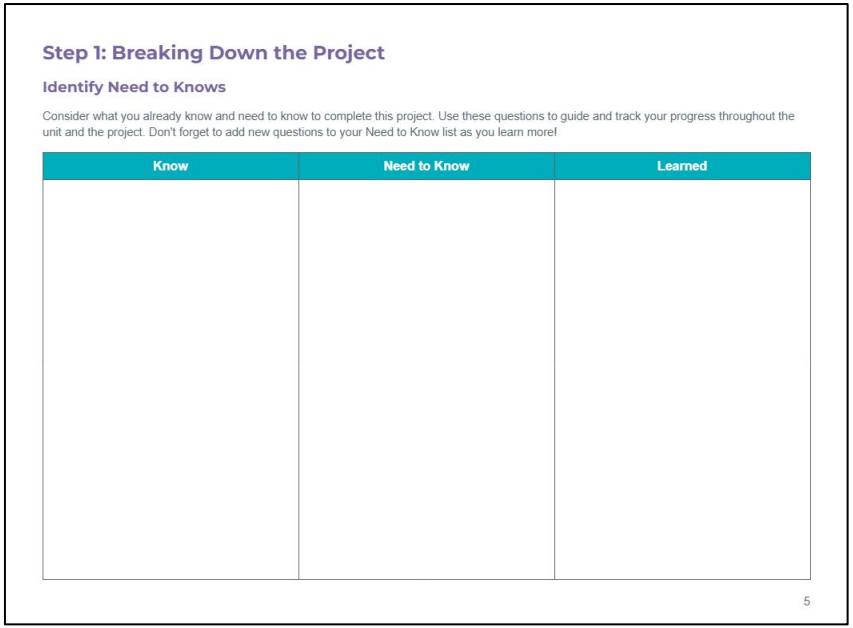
```
public static int calcSum(int num1, int num2)
```

```
public static int calcSum(int[] numbers)
```

✅ Do This:

Revisit your **Need to Knows**!

- Check off **answered questions** in the **Need to Know** column.
- Add what you have **learned** and **answers to any questions** in the **Learned** column
- Add any **new questions** to the **Need to Know** column



Step 1: Breaking Down the Project

Identify Need to Knows

Consider what you already know and need to know to complete this project. Use these questions to guide and track your progress throughout the unit and the project. Don't forget to add new questions to your Need to Know list as you learn more!

Know	Need to Know	Learned

5

Unit 7 - Lesson 4
Private Methods

Computer Science A

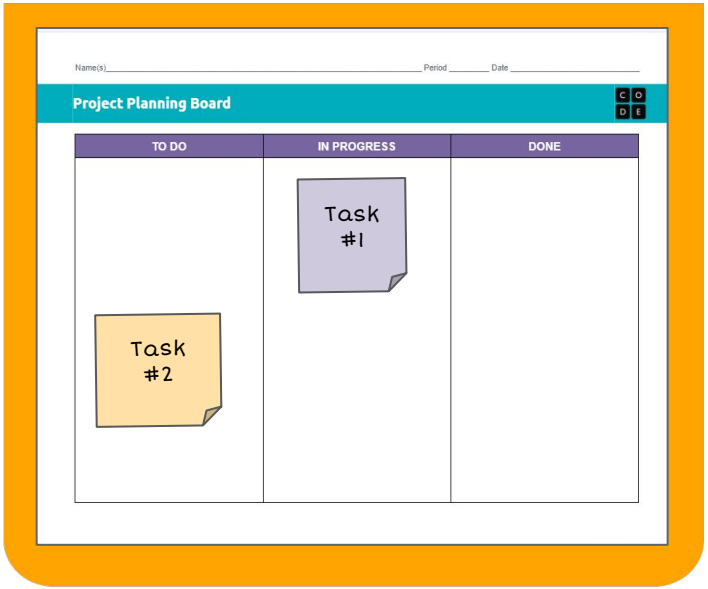
🎯 Benchmark #1: Due Lesson 6

- Brainstorm project ideas and goals
- Decompose the problem to identify the classes and methods you will need to implement
- Obtain and implement feedback from peers

✅ Do This:

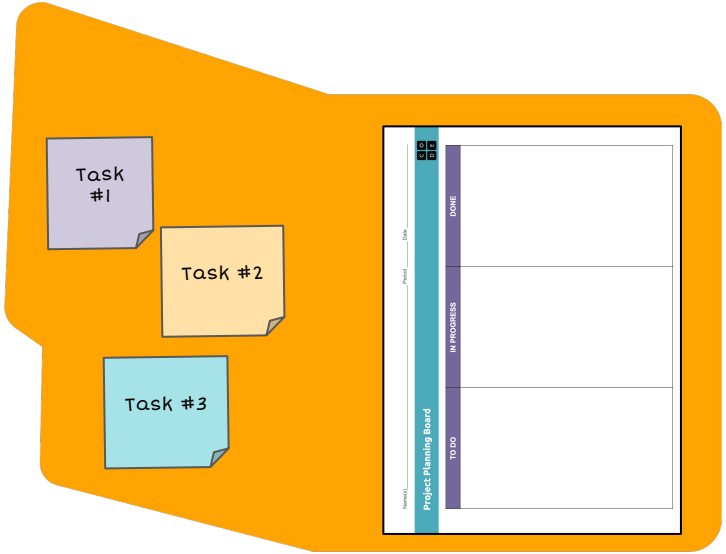
Move the task you will work on to the **IN PROGRESS** column of your Project Planning Board.

Work on your Creative Coding in The Theater Project.



✅ Do This:

Update your **Project Planning Board** and **Project Backlog** with any tasks you completed, changed, or added.



🚀 Question of the Day

How does making a method **`private`** change its functionality?

🎥 Private Methods

How are **`private`** methods similar and different from **`private`** instance variables?

Complete the guided notes on the 📝 **Unit 7 Guide**.

Dessert.java

```
public class Dessert {

   . . .

   private void giveDiscount() {
      price = price - 0.50;
   }

   . . .
}
```

MyConsole.java

```
. . .

myDessert.giveDiscount();

. . .
```

Console

```
/MyConsole.java:5: error: giveDiscount() has private access in
Dessert
    myDessert.giveDiscount();
              ^
```

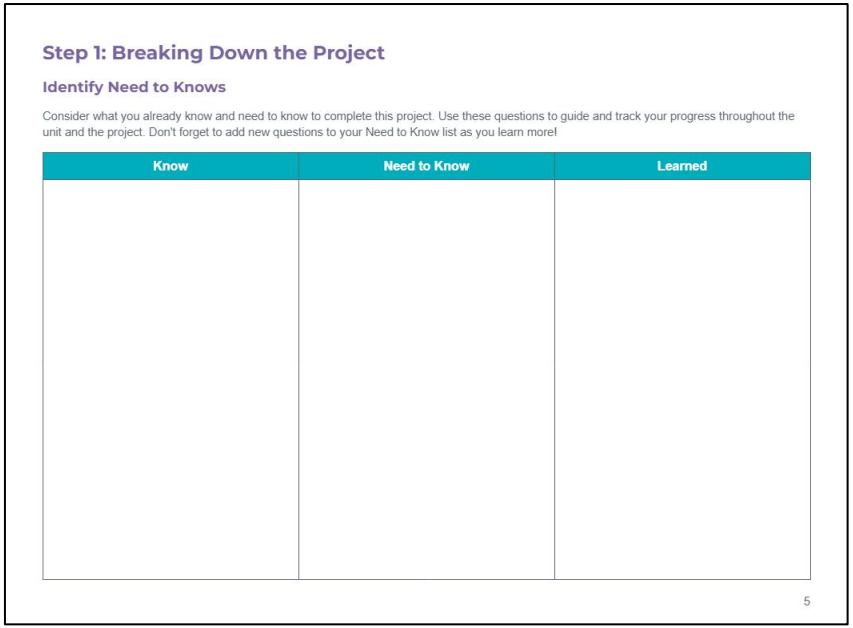
💡 Discuss:

Why would we want to use a **`private`** method?

✅ Do This:

Revisit your **Need to Knows**!

- Check off **answered questions** in the **Need to Know** column.
- Add what you have **learned** and **answers to any questions** in the **Learned** column
- Add any **new questions** to the **Need to Know** column



Step 1: Breaking Down the Project

Identify Need to Knows

Consider what you already know and need to know to complete this project. Use these questions to guide and track your progress throughout the unit and the project. Don't forget to add new questions to your Need to Know list as you learn more!

Know	Need to Know	Learned

5