Teacher's Surname	YILDIRIM	**Name:**	Fatma Merve
Title:	**Catch the bugs! A Game Design in Scratch**	**Time:**	2 hours
Subject:		**IT**	
Aims		**General competence**[1]: Awareness of computational thinking concepts to create a game where a frog character catches insects, applying basic computational thinking skills **Specific competence**[2]: Allowing students to deepen their understanding of a simple game creating. **Aim of the activity:** How to create a simple game via computational thinking	
Key CS elements:		Decomposition; Pattern recognition; Abstraction; Algorithm design.	
Age group :		**6-8 years old**	
Learning place:	**Çetin Şen Science and Art Center**	**Activity type:**	**extracurricular**
Materials: Computers or tablets with accsess to Scratch Projector or screen for demonstration Worksheets for planning game elements (Optional)		**Resources:** 1. Websites or educational videos related to Scratch.	

Learning development:

Problem definition:
Creating the right steps to create a simple game using computational thinking principles.

Introduction

1. Introduction to Computational Thinking

Discuss the Game Theme: Explain that today, students will create a game where a frog tries to catch butterflies, ladybugs, and beetles.

Introduce Key Concepts: Briefly introduce the computational thinking skills:

Decomposition: Breaking down the game into smaller parts.

Pattern Recognition: Identifying patterns in movements or behaviors of insects.

Abstraction: Focusing on essential details (e.g., basic movement rather than complex animations).

Algorithm Design: Planning step-by-step instructions to make the frog catch insects.

2. Planning the Game Elements

Decomposition: Identify each game component:

- Frog (main character)
- Butterfly, ladybug, and beetle (targets)
- Background (pond or grass)

Pattern Recognition: Recognize how each insect moves (e.g., butterflies fly, ladybugs walk slowly, beetles crawl).

Have students discuss in pairs how they might make each insect move differently to create patterns.

Abstraction: Simplify the design by focusing only on essential details:

- Frog only needs basic up, down, left, and right movement.
- Each insect requires unique yet simple movement patterns to distinguish them from each other.
- Keep score count and "catching" mechanics simple without additional animations.

Creating the Game in Scratch

- **Step 1:** Set Up Characters and Background
- Open Scratch and select a background (e.g., a grassy field or pond).
- Add a frog sprite as the main character and insect sprites (butterfly, ladybug, beetle).
- **Step 2:** Program Movements for Frog and Insects
- Frog: Use arrow keys to control frog movement. Introduce loops for continuous movement.
- Insects: Program each insect with different movement patterns:
- Butterfly: Flies in random directions.
- Ladybug: Moves slowly and turns occasionally.
- Beetle: Moves in a straight line and pauses.
- **Step 3:** Add a Catch Mechanism
- Program the frog to "catch" an insect when it touches them, making the insect disappear.
- Create a simple score counter that increases when an insect is caught.

Algorithm Design: Planning Game Mechanics

Explain Algorithm Design: Guide students to think step-by-step:

Step 1:Start with a "when green flag clicked" block to initialize the game.
Step 2: Program frog movements using arrow keys.
Step 3: Set insect movements using loops and "move" commands.
Step 4: Use conditional statements to detect when the frog touches an insect:
Increase score by 1.
Make the insect disappear or respawn in a different location.

Activity: Have students map out these steps in a simple flowchart or on paper before coding, ensuring clarity in their algorithms.

Testing, Debugging, and Refining

Testing: Encourage students to run the game, observing if the insects move correctly and if scoring functions as planned.
Debugging: Support students in identifying issues, such as if the frog isn't moving or if insects don't disappear when caught.
Refining with Abstraction: Ask students to review their code and remove any unnecessary blocks, focusing on efficient coding to keep the game functional but simple.

Wrap-Up and Reflection

- Discussion: Ask students to share what they learned about planning a game and using computational thinking.
- Reflection on Skills: How did decomposition, abstraction, pattern recognition, and algorithm design help them create their game?

Assessment:	Observe students' engagement with each computational thinking skill during the lesson and their ability to complete the game in Scratch. With abstraction and algorithm design, students focus on the game's essentials, creating a smoother workflow while learning computational thinking fundamentals. Let me know if you'd like to add further details!
Expected results:	By the end of the lesson, students will have created a functional Scratch game in which a frog catches various insects, demonstrating their understanding of decomposition, pattern recognition, abstraction, and algorithm design in computational thinking.
Notes: This project can also be linked to IT lessons, for grades 5,6 10-11 years-old students.	