# COSC 111: Computer Programming I

Dr. Bowen Hui
University of British Columbia Okanagan

## Last time

- Previously: What is computer software?
  - Course overview
  - Hardware vs. software
  - Software examples
- Today: From English to computer language

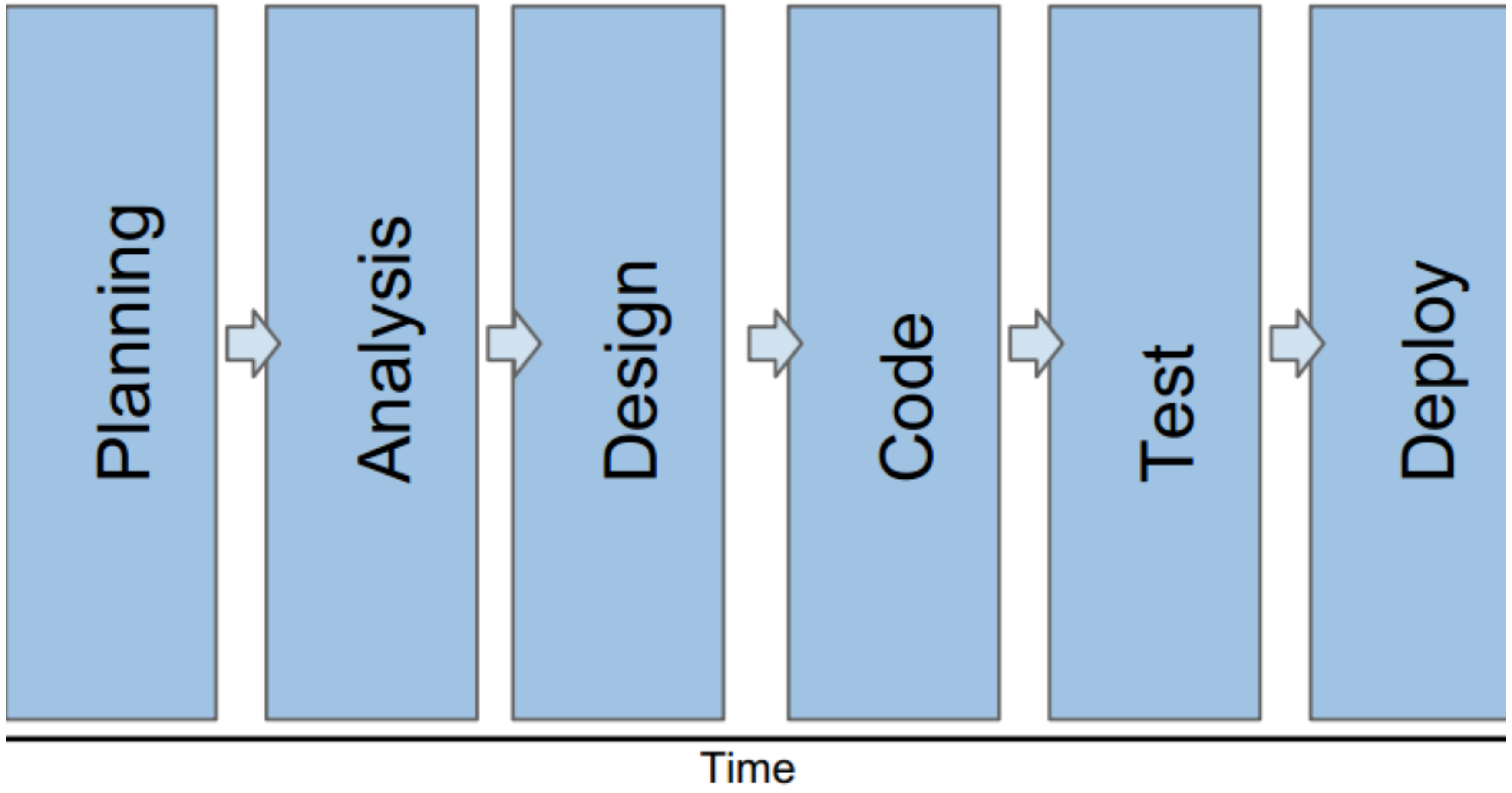
# Outline

- Part 1
  - Software process
  - Requirements
  - High level design
- Part 2
  - Java program structure
  - Attributes
  - Methods
    - Input-process-output (IPO) modeling
  - Testing a Java class
- Part 3
  - Example

## Software process

- Set of activities that produce software product
- Structured way of carrying out activities
- Goals:
  - Make clear steps to be done
  - Produce tangible work outputs
    - Design documents, decisions made, manuals
    - Software
    - Test scenarios
  - Allow for review of work by others

# Stages of software development



## Simple process for class assignments

- Read question
- Think about how to solve it for a couple minutes
- Solve it
- Get stuck – leave it for now
- Resume problem solving – not sure how you got there so rewrite everything
- Get a better attempt, but still stuck
- Stay up all night – mostly works
- Hand it everything (in case it’s relevant)
- Never see it again!

## A better process

- Read question – ask for clarification immediately
- Consider different approaches for solution
- Repeat until done:
  - Plan out pieces that need to be done
  - Work on solution for one part
  - Check correctness
  - Ensure consistency with other parts
- Submit completed and working solution happily

- Why is this better?

# Software development

- Basic activities:
  - Establishing the requirements
  - Creating a design
  - Implementing the code/software/program/app
  - Testing the implementation

## Software development

- Basic activities:
  - Establishing the requirements
    - High level planning
  - Creating a design
    - Detailed planning
  - Implementing the code
    - Coming up with solution
  - Testing the implementation
    - Making sure your solution works
- Best order to carry out these activities?

## Requirements

- Specify the “abilities” of a program
  - What it needs to be able to do
- How to get them?
  - English description from client
  - Analyze details, find comparisons, ask questions
  - Itemize software abilities (also English)

## Example client description

- Client says:

"I want to build a simple quiz game that lets people answer trivia questions. The questions should have multiple choice answers so it's not so hard for people if they get stuck. When the game is finished, the game will report a score."

- Where do we start?

## Group exercise (10 min)

- Client description:

"I want to build a simple quiz game that lets people answer trivia questions. The questions should have multiple choice answers so it's not so hard for people if they get stuck. When the game is finished, the game will report a score."

- Work in groups of 3.
- Create such a game using pen and paper.
- Cut up a piece of paper into 4 pieces so they serve as question cards. Have the question + multiple choice on one side, and the correct answer on the other side.

## Exercise review

- Demo a game?

# Exercise review

- How many questions?
- How many answers per question? Variant/constant?
- What type of questions?
- Can I play again?
- How to re-calculate the score?

- Very detailed
  - List all assumptions
  - Not a creative step in the process

# Requirements translation steps

1. Identify relevant nouns
   – Add requirements
2. Identify relevant adjectives
   – Add requirements
3. Come up with solution conceptually
   – Resolve unanswered questions
   – Add requirements

## Example client description

1. Underline nouns + circle relevant ones

"I want to build a simple quiz game that lets people answer trivia questions. The questions should have multiple choice answers so it's not so hard for people if they get stuck. When the game is finished, the game will report a score."

- Nouns tell us what needs to go into the program

# Example client description

1. Underline nouns + circle relevant ones

"I want to build a simple quiz game that lets people answer trivia questions. The questions should have multiple choice answers so it's not so hard for people if they get stuck. When the game is finished, the game will report a score."

- Nouns tell us what needs to go into the program

# Translating to requirements

- Quiz game
  - The software must be able to display a series of questions to the user
  - The software must be able to check if user inputted answer is correct
- Multiple choice answers
  - The software shall let users select from a set of possible answers for each question
- Score
  - The software shall tally the correct answers and display a summary score at the end of a game

# Example client description

2. Underline adjectives + circle relevant ones

"I want to build a simple quiz game that lets people answer trivia questions. The questions should have multiple choice answers so it's not so hard for people if they get stuck. When the game is finished, the game will report a score."

- Adjectives tell us how the users should perceive the software after it has been built

## Translating to requirements

- Simple
  - The software must make it obvious what the user needs to do at each step
- Is finished
  - The software shall notify users when a game is finished

- Are we done?

3. What else might we want to know if we imagine building a pen-and-paper game?

## Questions that arise

- Who enters the quiz questions?
- How many questions?
- How many multiple choice answers per question?
- Do questions need to be numbered?
- Do answers need to be numbered?
- What kind of scoring is reported?
- When a game ends, do we start over?
- Will the score be maintained across games?
- Are there multiple players?
- Will scores be compared across players?

## New requirements to add

- Who enters the quiz questions?
  - The software shall let the user enter the questions, multiple choice answers, and the correct answer.
- How many questions?
- How many multiple choice answers per question?
- Do questions need to be numbered?
- Do answers need to be numbered?
- What kind of scoring is reported?
- When a game ends, do we start over?
- Will the score be maintained across games?
- Are there multiple players?
- Will scores be compared across players?

## New requirements to add

- Who enters the quiz questions?
- How many questions?
  - The software must pose 10 questions per game.
- How many multiple choice answers per question?
- Do questions need to be numbered?
- Do answers need to be numbered?
- What kind of scoring is reported?
- When a game ends, do we start over?
- Will the score be maintained across games?
- Are there multiple players?
- Will scores be compared across players?

# New requirements to add

- Who enters the quiz questions?
- How many questions?
- How many multiple choice answers per question?
  - The software shall have 4 multiple choice answers for all questions.
- Do questions need to be numbered?
- Do answers need to be numbered?
- What kind of scoring is reported?
- When a game ends, do we start over?
- Will the score be maintained across games?
- Are there multiple players?
- Will scores be compared across players?

# New requirements to add

- Who enters the quiz questions?
- How many questions?
- How many multiple choice answers per question?
- Do questions need to be numbered?
  - *… continue process*
- Do answers need to be numbered?
- What kind of scoring is reported?
- When a game ends, do we start over?
- Will the score be maintained across games?
- Are there multiple players?
- Will scores be compared across players?

## Next step

- So far: client description -> requirements
- Next: requirements -> high level design
- High level design
    1. Which objects need to be modeled?
    2. What is the main responsibility of each object?
    3. What do these objects need to remember?
    4. What do these objects need to be able to do?

## Quiz game nouns

- Which objects need to be modeled?
- Start with the relevant nouns:
  - Quiz game
  - Multiple choice answers (for questions)
  - Score (for player)

# Quiz game design

- Game
  - Main responsibility:
  - Needs to keep track of:
  - Have ability to do:
- Question
  - Main responsibility:
  - Needs to keep track of:
  - Have ability to do:
- Player
  - Main responsibility:
  - Needs to keep track of:
  - Have ability to do:

## Part 1 summary

- From client description to requirements
  - Identify relevant nouns and adjectives
  - Conceptualize solution to refine requirements
- From requirements to high level design
  - Which objects need to be modeled?
  - Model details
    1. Main responsibility:
    2. Needs to keep track of:
    3. Have ability to do:

# Once you have a design…

1. Translate each kind of object into a Java class
2. Translate group of things to keep track of into attributes
3. Translate each ability into a skeleton method
   - Detail the list of commands to issue in each method
   - Rule of thumb: each ability corresponds to one process (or routine, or method, or…)
     - Later, you will start to combine processes together
     - Later, you will start to decompose complicated processes into multiple, smaller processes

# From Planning to Implementation

- Quiz game design
  - Keep track of
  - Abilities

```
class Game
{
      // attributes
      // methods
}
```

- Test design??

```
class TestQuiz
{
      Game quiz = new Game();
      quiz.startGame();
      // continue here
}
```

# From Planning to Implementation

- Quiz game design
  - Keep track of
  - Abilities

- Test design??

*Capital letter*

```
class Game
{
      // attributes
      // methods
}
class TestQuiz
{
      Game quiz = new Game();
      quiz.startGame();
      // continue here
}
```

*Simplified template*

## Classes for Game, Question, Player

```
class Game
{
    // attributes
    // methods
}
class Question
{
    // attributes
    // methods
}
```

```
class Player
{
    // attributes
    // methods
}
```

# Classes for Game, Question, Player

```
class Game
{
    // attributes
    // methods
}
class Question
{
    // attributes
    // methods
}
```

```
class Player
{
    // attributes
    // methods
}
```

Braces have to match

This is a comment

# Attributes

- Type of information

| | |
|---|---|
| – Number? | int x; |
| – Phrase? | String x; |
| – Truth value? | boolean x; |
| – Other? | *may create your own* |
| | Player bob; |

## Attributes

- Type of information

| | |
|---|---|
| – Number? | int x; |
| – Phrase? | String x; |
| – Truth value? | boolean x; |
| – Other? | *may create your own* |
| | Player bob; |

- Amount of information

| | |
|---|---|
| – Single piece? | int x; |
| | String x; |
| | Player bob; |
| – Sequence? | int[] x; |
| | String[] x; |
| | Player[] challengers; |

## Giving attribute values

- Initialization = happens for first time
- Two alternatives:
  - int num = 5;
  - int num;
    num = 5;

This alone is called a declaration

- Don't have both for the same variable!
- More examples:
  - String name = "bob";
  - boolean searchResult;
    searchResult = false;

# Changing attribute values

- Once you've declared a variable, you can give it values and change it as needed
- Good example:

  -
    ```
    int num;
    num = 5;
    // do some stuff in between
    num = 9;
    ```
- Bad example:

  -
    ```
    int num = 5;
     num = "bob is my neighbour";
    ```
  - Type of information cannot change

## Methods

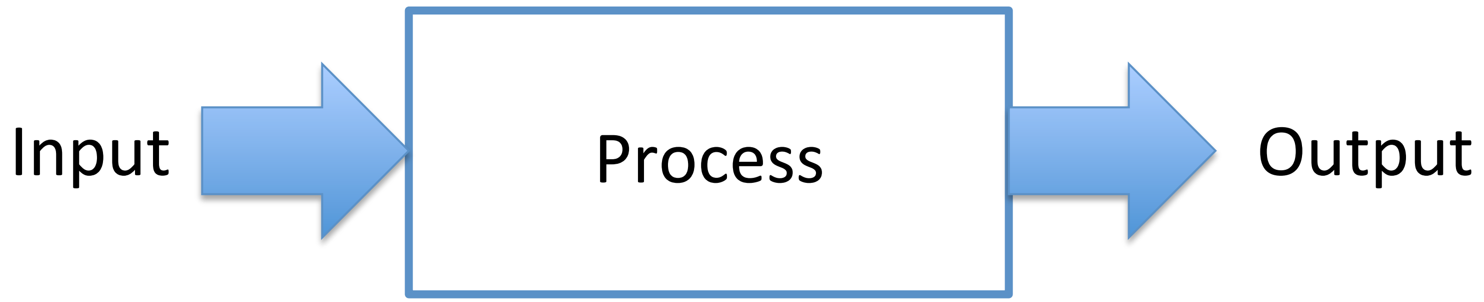
- Think of it as a routine
  - A set of repetitive steps grouped together
  - How to do something
- Used to define an object's ability
- Examples in your everyday life?

## Methods

- Think of it as a routine
  - A set of repetitive steps grouped together
  - How to do something
- Used to define an object's ability
- Examples in your everyday life?
  - Carry conversation
  - Walking to a classroom
  - Cooking spaghetti
  - Doing homework

## Input-process-output (IPO)

- Model to show process interaction
- Input – information, resources needed
- Process – step-by-step commands
- Output – results of transforming I by P

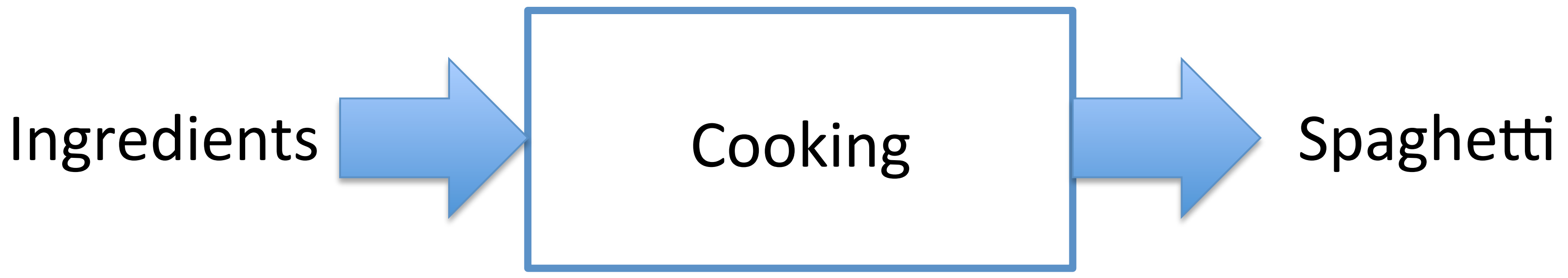


# Input-process-output (IPO)

- Analogy: recipe
  - Input = ?
  - Process = ?
  - Output = ?

# Input-process-output (IPO)

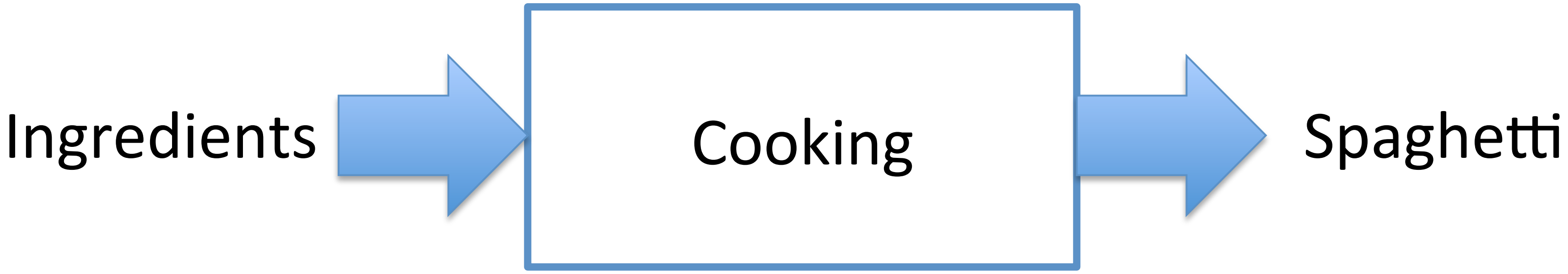
- Analogy: recipe
  - Input = ?
  - Process = ?
  - Output = ?



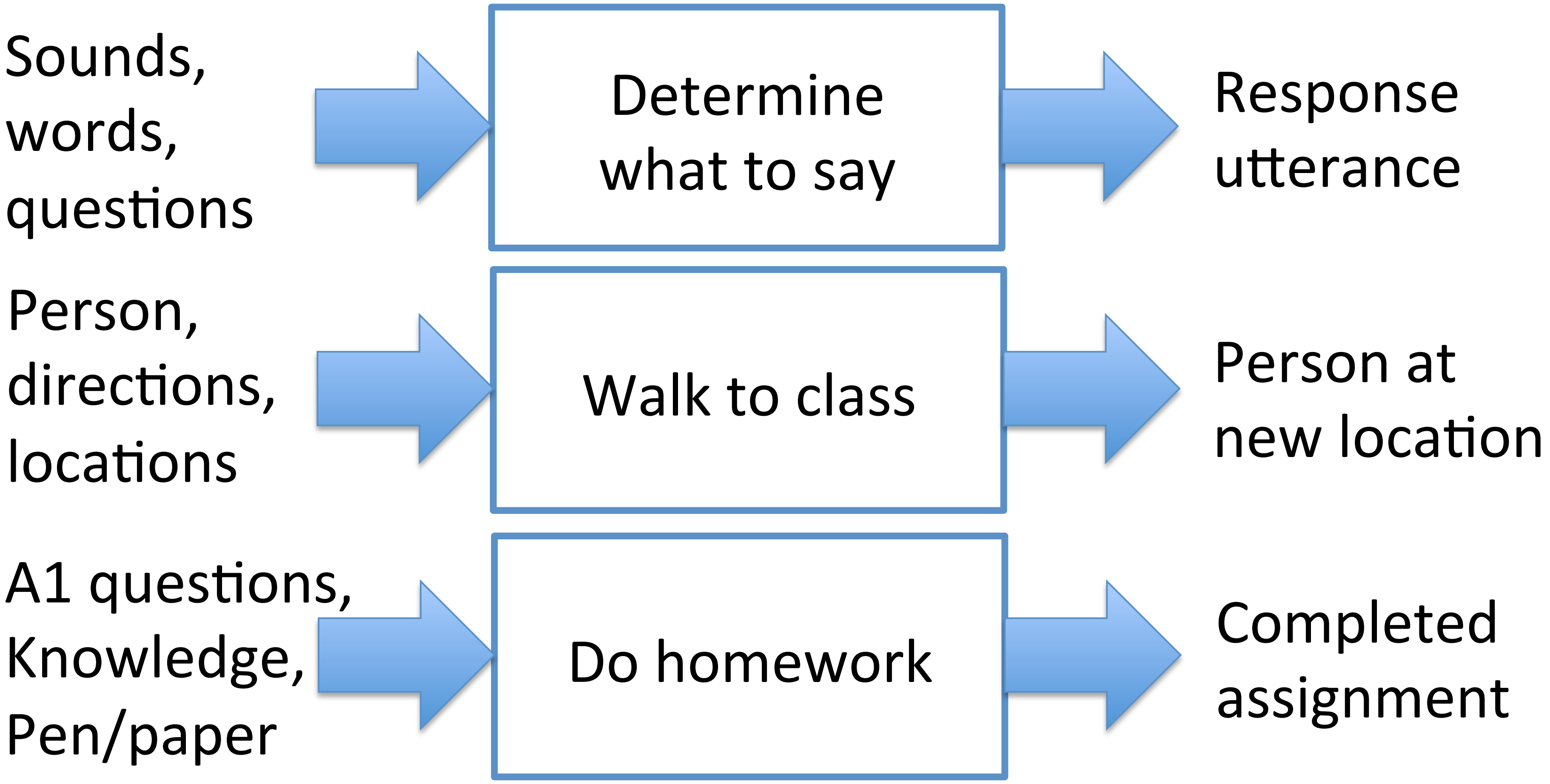
## Input-process-output (IPO)

- Analogy: recipe
  - Input = ?
  - Process = ?
  - Output = ?

Looks like a function?
Consider: 1 + 2 = 3



# IPO Examples



# Back to quiz game example

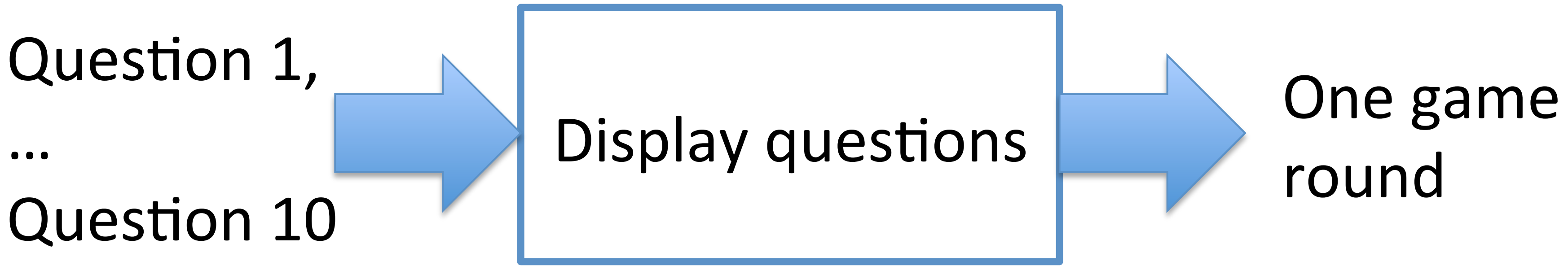
- Game ability
  - Display 10 questions in each game
- Question ability
  - Set correct answer
- Player ability
  - Maintain own high score across all previous games

# Back to quiz game example

- Game ability
  - Display 10 questions in each game
- IPO?

# Back to quiz game example

- Game ability
  - Display 10 questions in each game

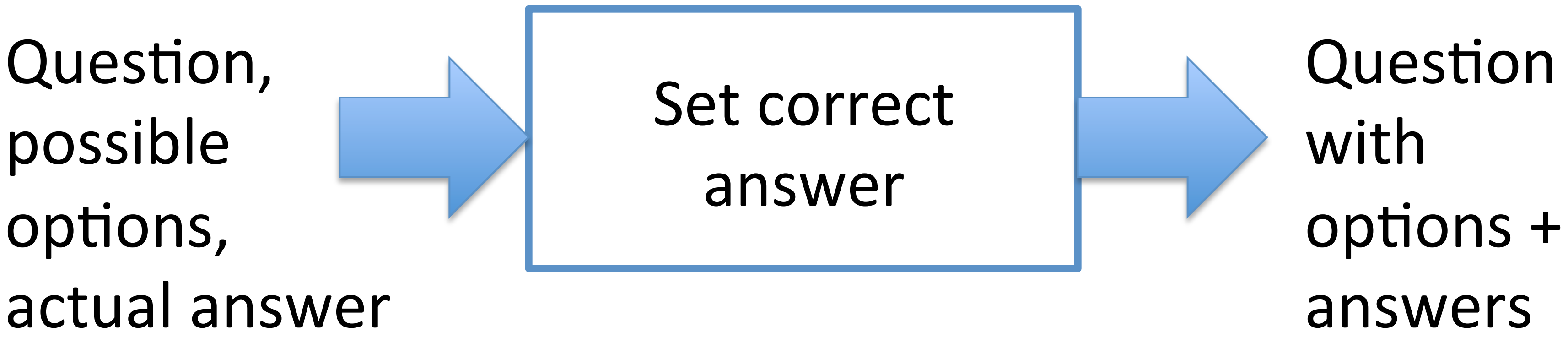


## Back to quiz game example

- Question ability
  - Set correct answer
- IPO?

## Back to quiz game example

- Question ability
  - Set correct answer

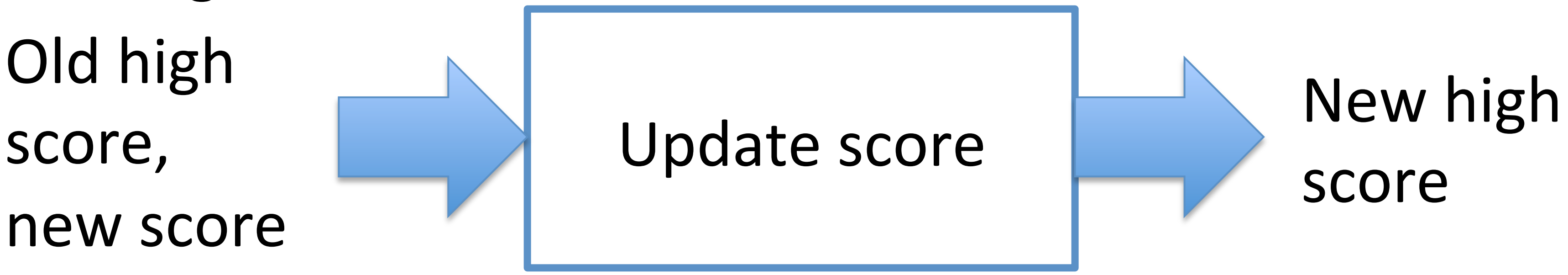

Question, possible options, actual answer
Set correct answer
Question with options + answers

# Back to quiz game example

- Player ability
  - Maintain own high score across all previous games
- IPO?

# Back to quiz game example

- Player ability
  - Maintain own high score across all previous games



## Inside the process?

- What goes into the process?
  - List of commands
  - Steps to transform I to O
  - For now: (mostly) leave blank

# Translating to Java methods

- Template structure:

  returnType methodName( varType$_1$ var$_1$, …, varType$_k$ var$_k$ )  
  {  
      // commands go here  
  }

# Translating to Java methods

- Template structure:

  $\text{returnType methodName( }\overbrace{\text{varType}_1\ \text{var}_1,\ \ldots,\ \text{varType}_k\ \text{var}_k}^{\text{Input(s)}}\text{ )}$

  {

          // commands go here

  }

## Translating to Java methods

- Template structure:

$$\text{returnType methodName( }\overbrace{\text{varType}_1\ \text{var}_1,\ \ldots,\ \text{varType}_k\ \text{var}_k}^{\text{Input(s)}}\text{ )}$$

$$\boxed{\begin{array}{l}\{\\ \qquad \text{// commands go here}\\ \}\end{array}}\quad \text{process}$$

## Translating to Java methods

Input(s)

- Template structure:

output

returnType methodName( $\text{varType}_1$ $\text{var}_1$, …, $\text{varType}_k$ $\text{var}_k$ )

```
{
    // commands go here
}
```

process

# Translating to Java methods

- Template structure:

output $\rightarrow$ returnType methodName( $\overbrace{\text{varType}_1 \text{ var}_1, \ldots, \text{varType}_k \text{ var}_k}^{\text{Input(s)}}$ )

```
{
        // commands go here
}
```

process

- Parameters = input variables inside brackets
  - Like a math function e.g. f(x)
  - These vars can be used in the commands inside method
- Return type and parameter type must follow the same rules as the type of information for attributes

## Method with no parameters

- Template structure:

```
returnType methodName( varType1 var1, …, varTypek vark )
{
    // commands go here
}
```

- Change k=0:

```
returnType methodName()
{
    // commands go here
}
```

# Method with something to return

- Template structure:

```
returnType methodName( varType_1 var_1, …, varType_k var_k )
{
        // commands go here
}
```

- Specify return type of the actual type of information to be returned:

```
int methodName( varType_1 var_1, …, varType_k var_k )
{
        // commands go here
        return -1;
}
```

  - Must have return statement

## Method with nothing to return

- Template structure:

  returnType methodName( varType$_1$ var$_1$, …, varType$_k$ var$_k$ )
  {
  // commands go here
  }

- Specify return type as follows:

  void methodName( varType$_1$ var$_1$, …, varType$_k$ var$_k$ )
  {
  // commands go here
  }

  – Don’t need any return statement

# Special method: constructors

- Template structure:

  returnType methodName( varType$_1$ var$_1$, …, varType$_k$ var$_k$ )<br>
  {<br>
          // commands go here<br>
  }

- Remove return type as follows:

  methodName( varType$_1$ var$_1$, …, varType$_k$ var$_k$ )<br>
  {<br>
          // commands go here<br>
  }

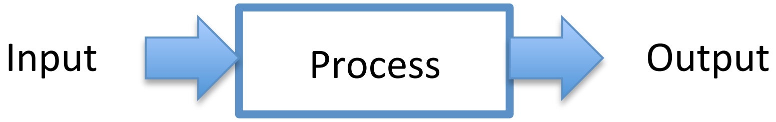
  – Don't need any return statement

# Part 2 summary

- Class template

```
class Game
{
        // attributes
        // methods
}
```

- Attributes
  - Number, phrase, truth value, or other
  - Single or sequence
- IPO:



- Method template:

returnType methodName( varType$_1$ var$_1$, …, varType$_k$ var$_k$ )
```
{
        // commands go here
}
```

## Example design

```
class Game
{
    // attributes
    // methods
}
```

# Example design

```
class Game
{
    // attributes
    int num;
    Player person;
    Question[] questionSet;

    // methods
}
```

# Example design

```
class Game
{
    // attributes
    int num;
    Player person;
    Question[] questionSet;

    // constructor – always initialize attributes in here
    Game()
    {
        num = 5;
        person = new Player();
        questionSet = new Question[ num ];
    }

    // other methods here
}
```

## Example design

```
class Quiz
{
        // attributes …
        // constructor …
        // other methods here
        void startGame() { … }
        int getPlayerScore()
        {
                // access score in
                // person’s attributes
                return -1;
        }
        int compHighs( int score1, int score2 )
        {
                // compare score1 with score2
                // return higher of the two
                return -1;
        }
        // continue next column
```

```
        boolean isGameOver()
        {
                // are all questions answered?
                // if so, return true
                return false;
        }
        String isBetter( Player p2 )
        {
                // compare my person’s
                // high score to p2’s highscore
                // return name of my person
                // if it has higher score
                // else return p2’s name
                return “temp”;
        }
}
```

# Testing the classes

```
class TestQuiz
{
    Player bob = new Player( “robert” );
    // call its methods here

    Question q1 = new Question( “What is …”, “a. 1”, “b. 5”, “c. 10” );
    q1.setRealAnswer( “b” );
    // call its methods here

    Game quiz = new Game();
    quiz.startGame();
    // call its methods here
}
```

Calls the constructor method

Calls a method
pass in real values

# Part 3 summary

- Steps:
  - Convert class design to class skeleton
  - Convert each thing to keep track of into attributes
    - Declare them only
  - Build a constructor method
    - Initialize all the attributes in it
  - Convert each class ability into a method
  - Create a test class by creating object and calling its methods

```
class TestQuiz
{
        Quiz game = new Quiz();
        game.startGame();
}
```

## Next class

- Practice on today's concepts

- Administration:
  - A0 due 10:59pm
  - Don't forget the readings and lab prep