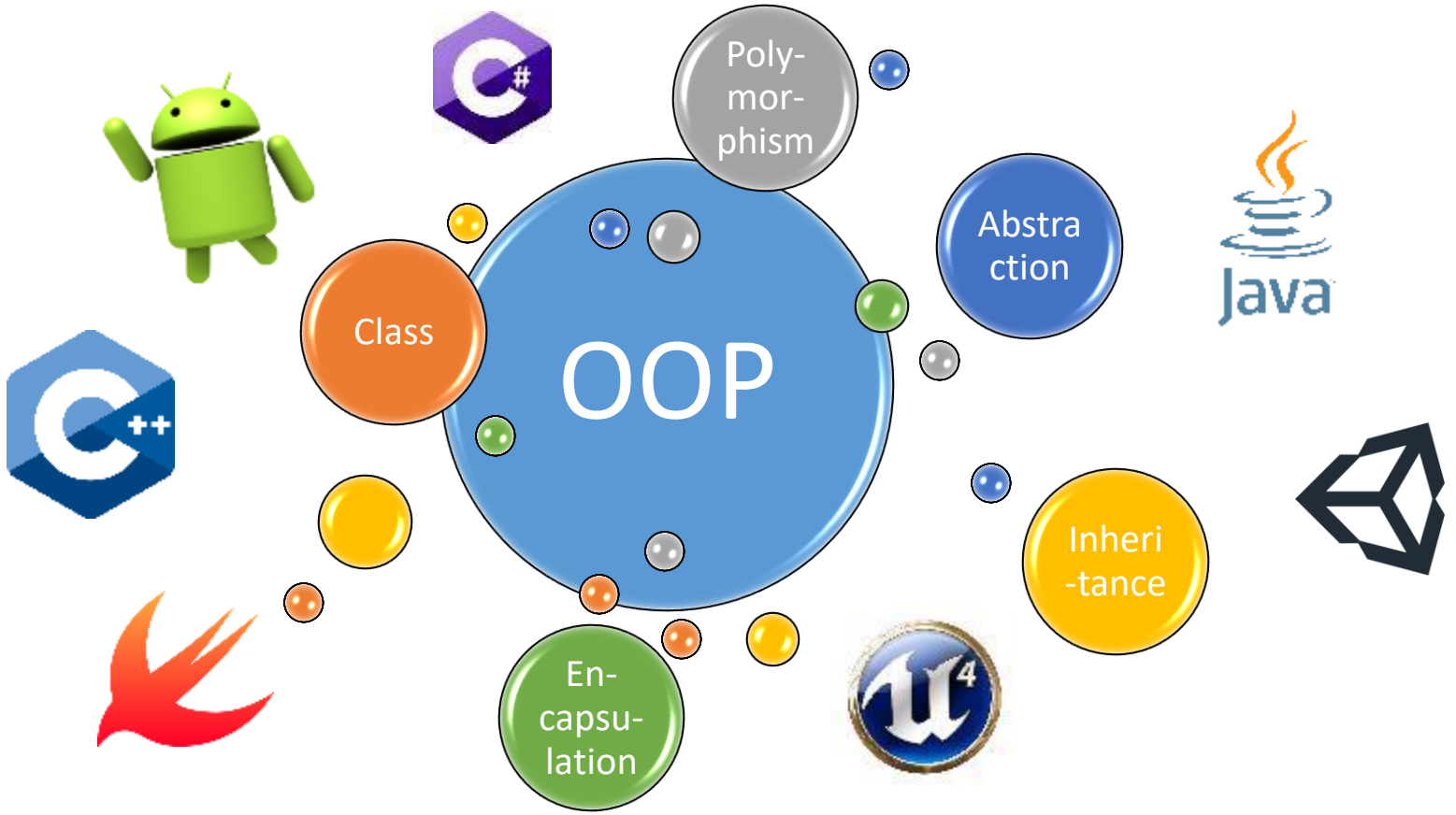
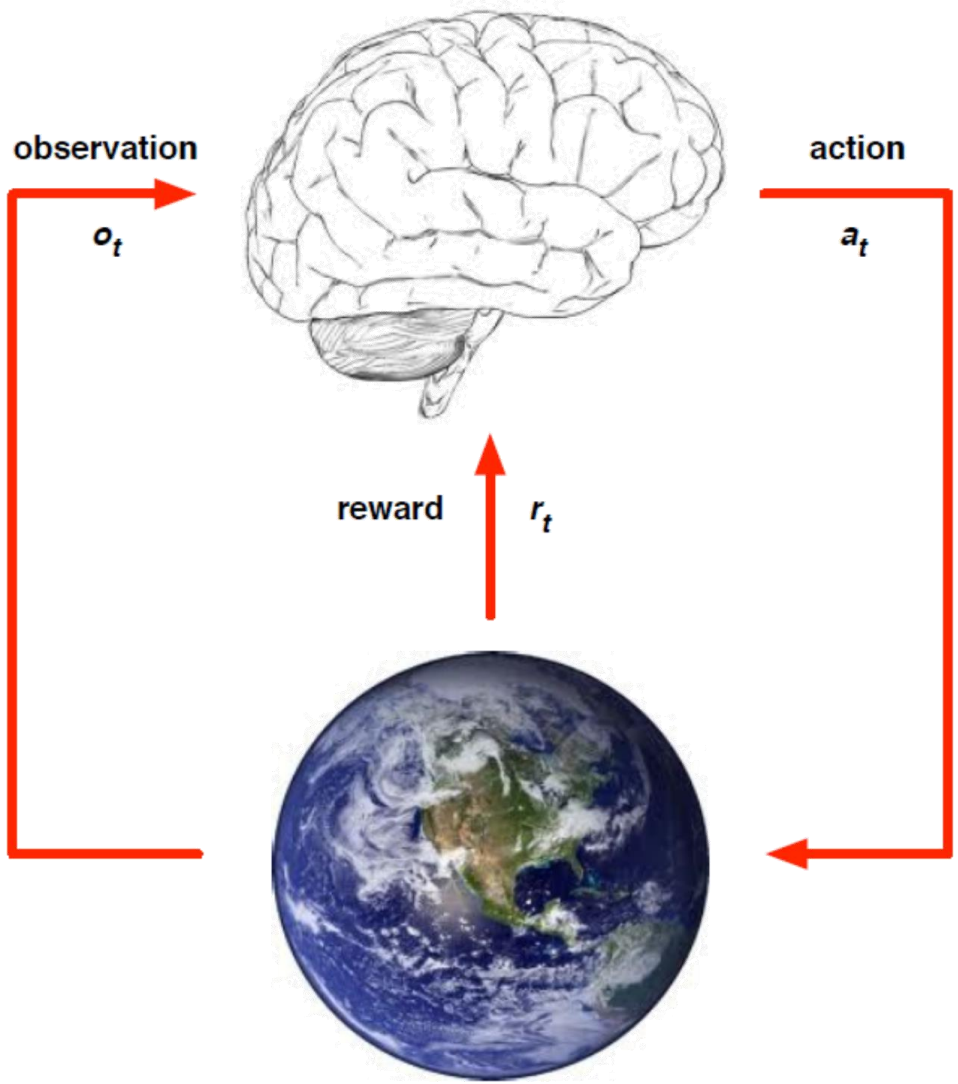


Learning to Play Tetris via Deep Reinforcement Learning

Kuan-Ting Lai

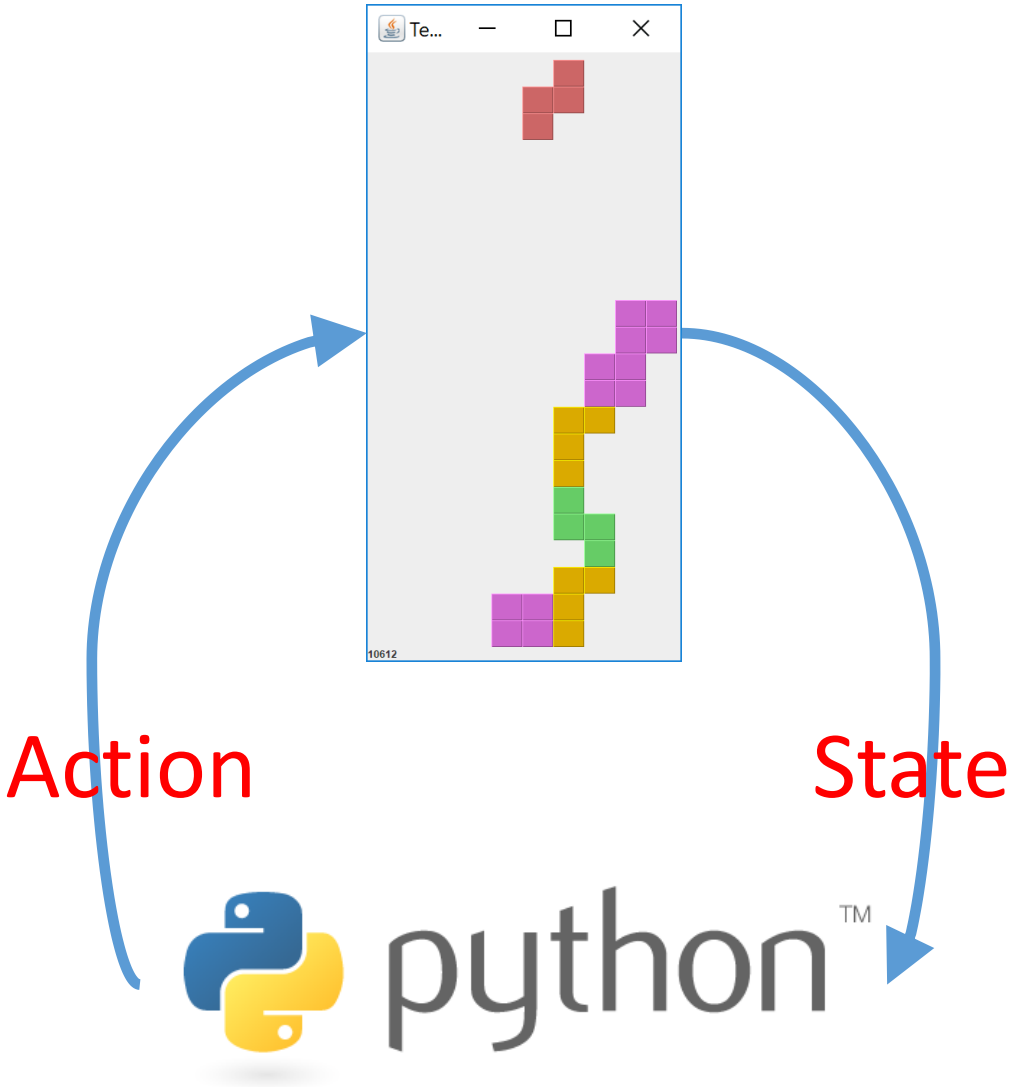
2020/5/25

Deep Reinforcment Learning (DRL)



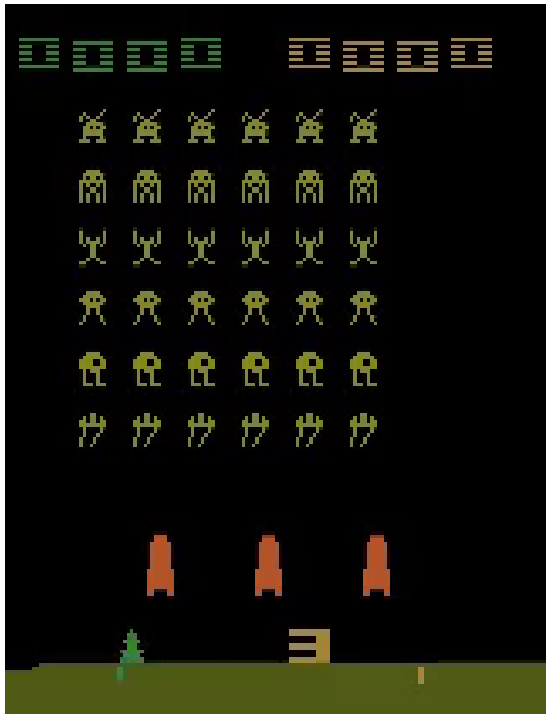
Establishing Communiation between Java and Python

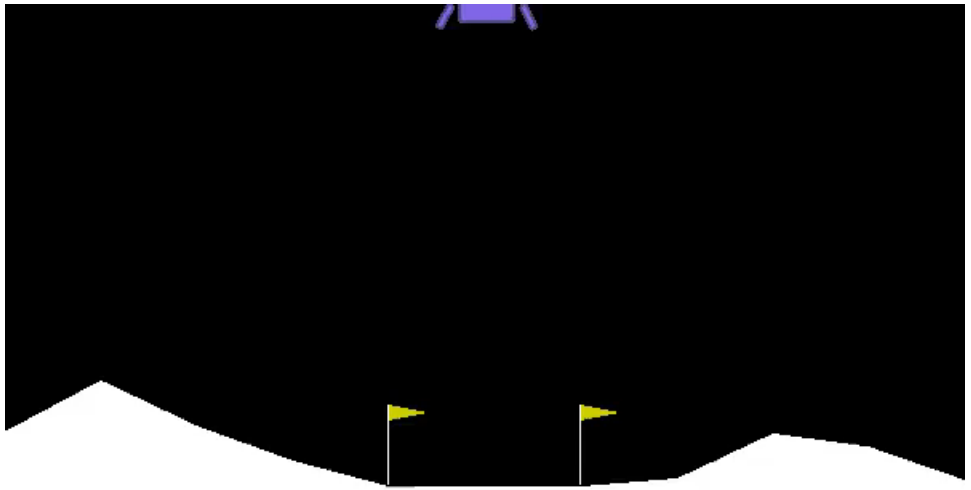
- Remote Procedure Call (RPC)
- MsgPack RPC
  - MessagePack: It's like JSON, but fast and small https://msgpack.org/index.html

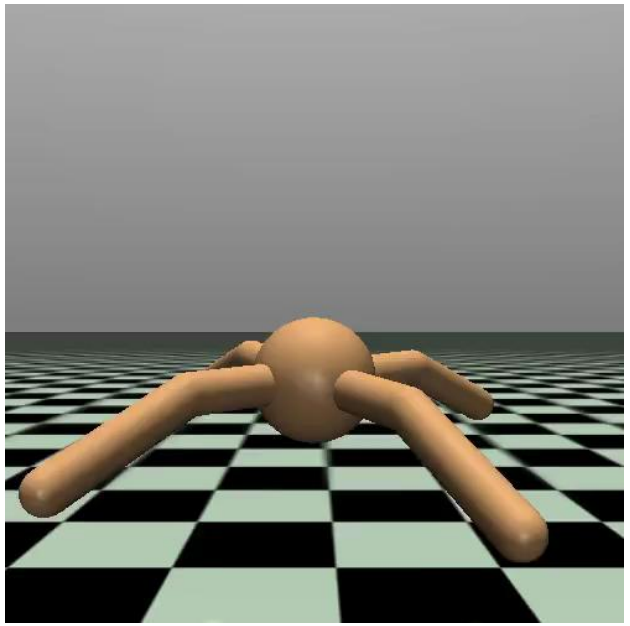


OpenAI Gym

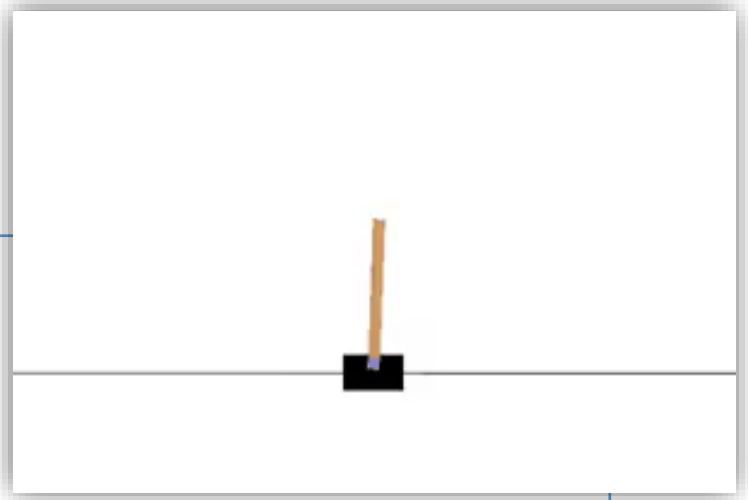
- https://gym.openai.com/







Hello Gym Example

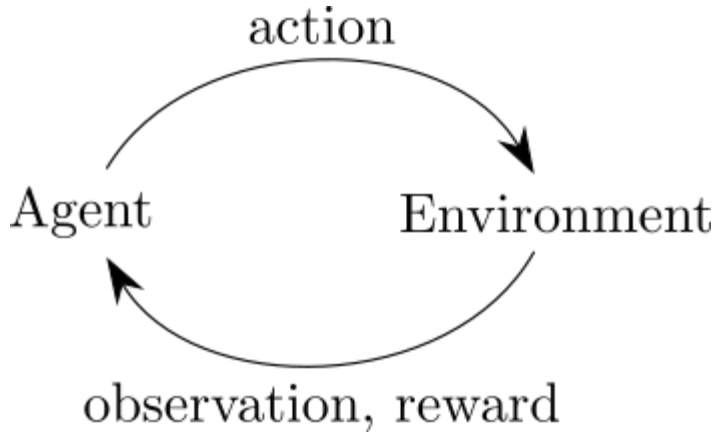


```
import gym
env = gym.make("CartPole-v1")
observation = env.reset()
for _ in range(1000):
  env.render()
  # your agent here (this takes random actions)
  action = env.action_space.sample()
  observation, reward, done, info = env.step(action)

  if done:
    observation = env.reset()

env.close()
```

https://gym.openai.com/docs/

step()

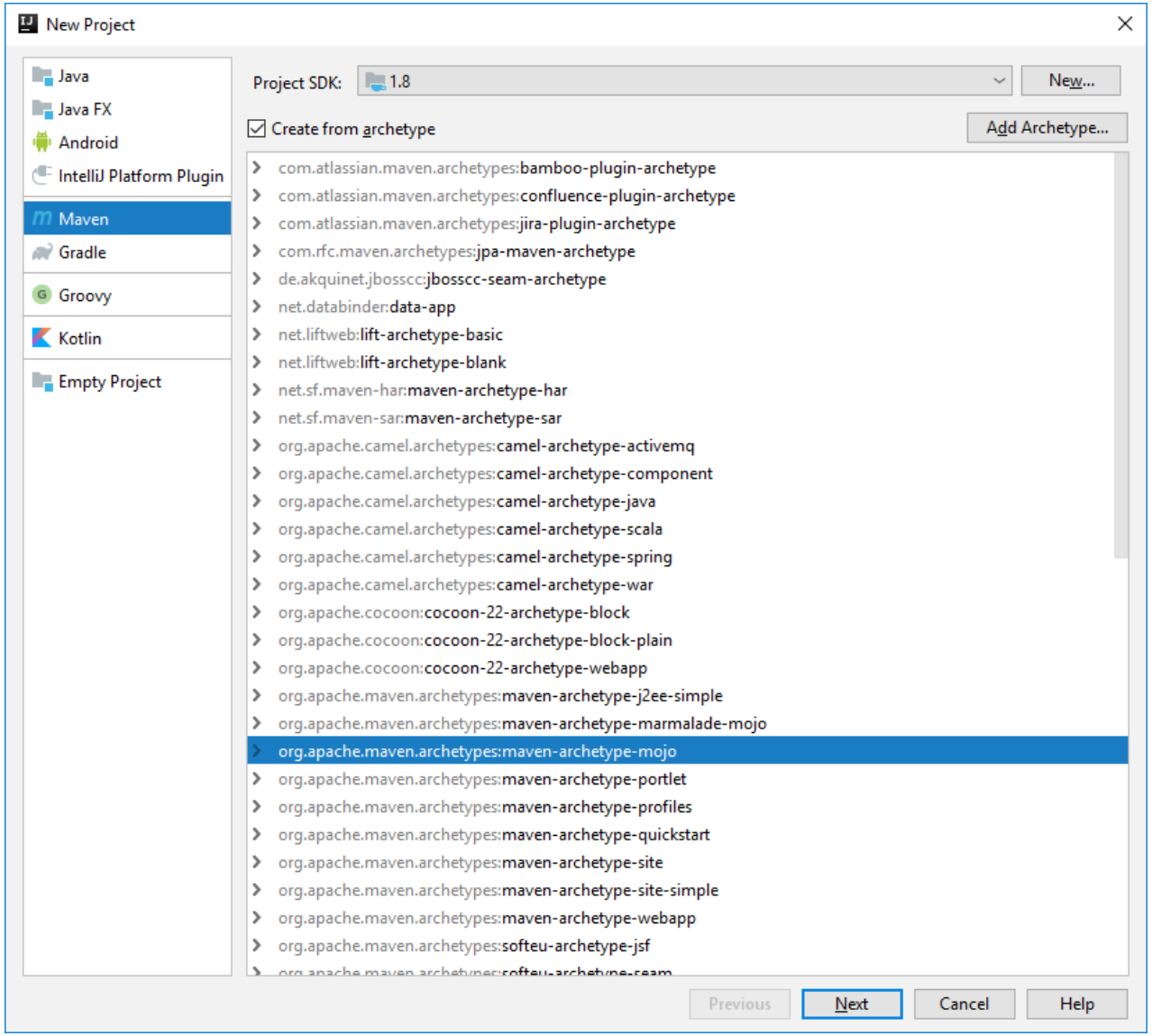
- observation (state S_t):
  - Observation of the environment. Ex: pixel data from a camera, joint angles and joint velocities of a robot, or the board state in a board game.
- reward (r_t)
  - amount of reward achieved by the previous action.
- done:
  - True indicates the episode has terminated
- info:
  - diagnostic information useful for debugging.



Creating Tetris DRL Project

Creating a Maven Project

- New Project -> Maven
  - GroupId: org.aiotlab
  - ArtifactId: Tetris4DRL

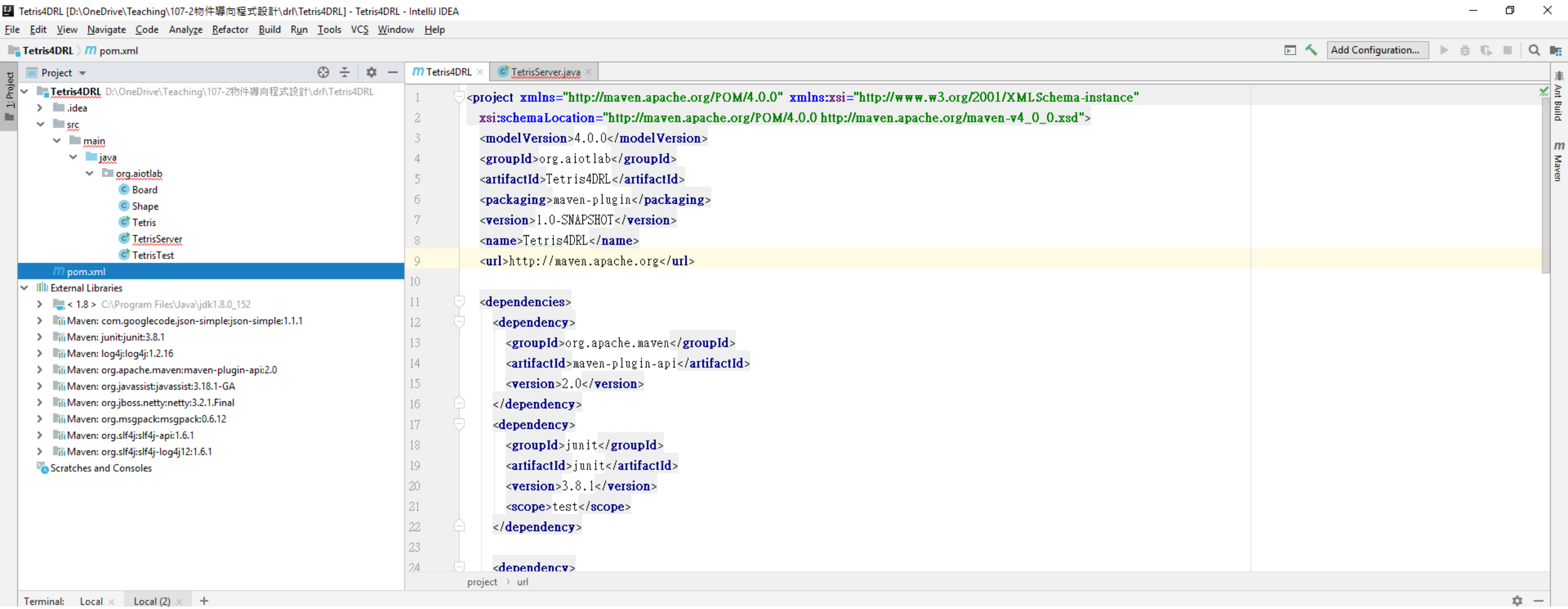


Adding Dependencies to “pom.xml”

- Add the dependencies of msgpack-rpc-java to pom.xml

```
<dependencies>
    <dependency>
        <groupId>org.msgpack</groupId>
        <artifactId>msgpack</artifactId>
        <version>0.6.12</version>
    </dependency>

    <dependency>
        <groupId>org.jboss.netty</groupId>
        <artifactId>netty</artifactId>
        <version>3.2.1.Final</version>
        <exclusions>
            <exclusion>
                <groupId>javax.servlet</groupId>
                <artifactId>servlet-api</artifactId>
            </exclusion>
            <exclusion>
                <groupId>commons-logging</groupId>
                <artifactId>commons-logging</artifactId>
            </exclusion>
        </exclusions>
    </dependency>

    <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-api</artifactId>
        <version>1.6.1</version>
    </dependency>

    <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-log4j12</artifactId>
        <version>1.6.1</version>
    </dependency>
</dependencies>
```

Reimporting Maven



MsgPack-RPC for Java

Download and Compile msgpack-rpc-java

1. Install Maven (link)
2. Download msgpack-rpc-java
   – https://github.com/msgpack-rpc/msgpack-rpc-java
3. Compile source code
   – mvn compile
4. Create jar file
   – mvn package
5. Fix a bug “type optional is ambiguous” (link)
6. Copy and add the library to IntelliJ project
   – Add “target/msgpack-rpc-0.7.1-SNAPSHOT.jar” to our project

Fixing Error “type optional is ambiguous”

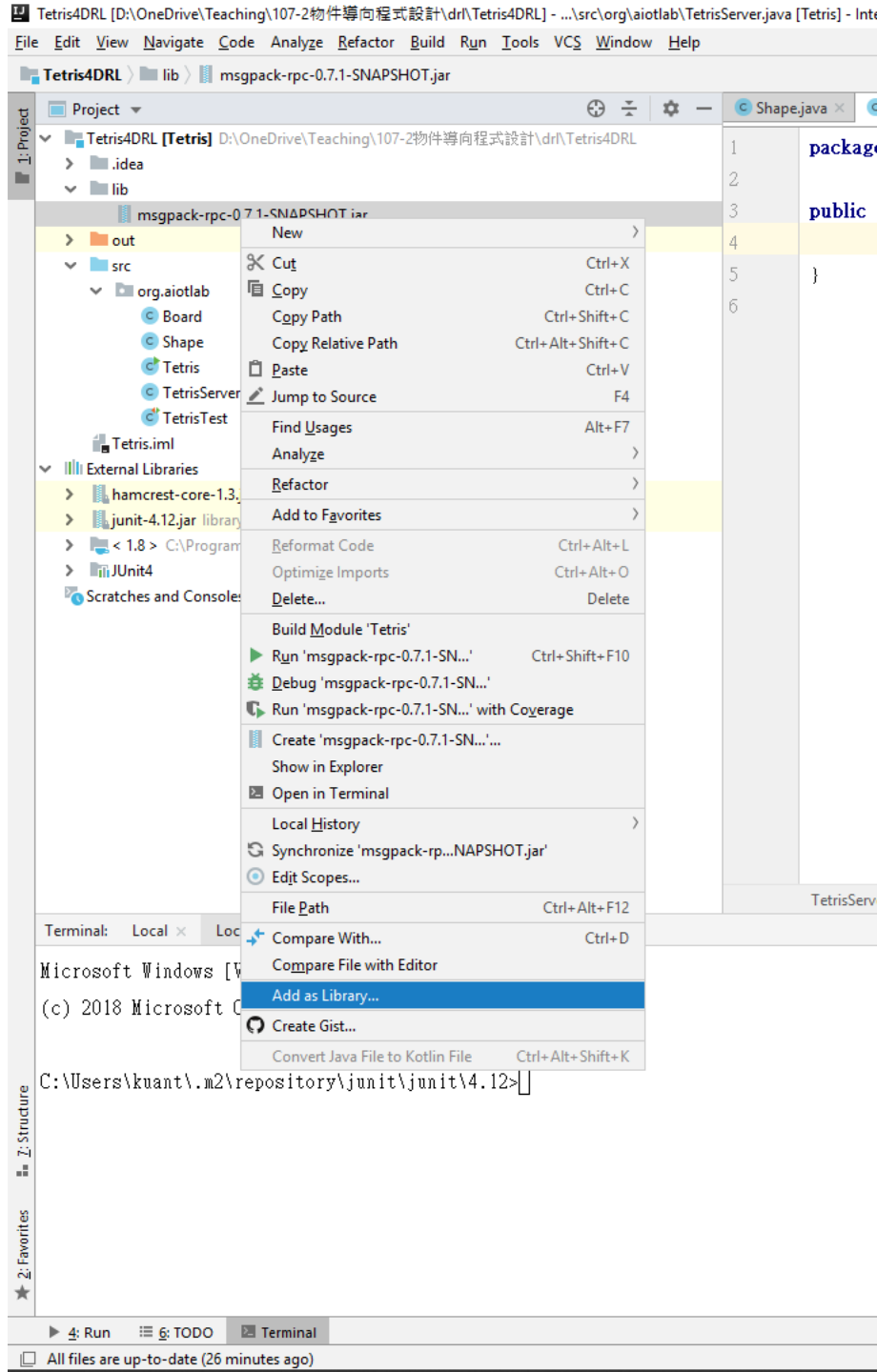
- https://github.com/msgpack-rpc/msgpack-rpc-java/pull/12

- Issue: Maven can't compile the package because of ambiguous optional type in AnnotationTest

- Solution: Explicitly Import org.msgpack.annotation.Optional

Add “msgpack-rpc-x.x.x.jar”

- Create a folder “lib”
- Copy and paste “msgpack-rpc-0.7.1.jar” to the lib/ folder
- Right click the jar file, and select “Add as Library…”



Remove Timer in Board.java

```
private void initBoard(Tetris parent) {
    setFocusable(true);
    /*timer = new Timer();
            timer.scheduleAtFixedRate(new ScheduleTask(),
            INITIAL_DELAY, PERIOD_INTERVAL);*/
    ...
}
```

```
private void newPiece() {
    ...
    if (!tryMove(curPiece, curX, curY)) {
        curPiece.setShape(Tetrominoe.NoShape);
        //timer.cancel();
        isStarted = false;
        statusbar.setText("Game over");
    }
}
```

Create the RPC Server

```
public class TetrisServer
{
    public static void main(String[] args) {

        // Create a RPC server
        EventLoop loop = EventLoop.defaultEventLoop();
        Server svr = new Server();
        svr.serve(new TetrisServer());
        try {
            svr.listen(server_port);
            System.out.println("Tetris RPC server is listening at " + server_port);
            loop.join();
        } catch (IOException exp1) {
            exp1.printStackTrace();
        } catch (InterruptedException exp2) {
            exp2.printStackTrace();
        }
    }
}
```

Add OpenAI Gym API

```
public class TetrisServer {
    static int server_port = 10612;
    static Tetris game;
    // OpenAI GYM API
    public int [] reset() {
        game.restart();
        return game.getBoardState();
    }
    public int [] step(int action_type) { return game.step(action_type); }
    public int getTotalReward() { return game.getLinesRemoved(); }
    public boolean isDone() { return game.isGameOver(); }

    public static void main(String[] args) {
        game = new Tetris();
        game.setVisible(true);

        // Create a RPC server
        EventLoop loop = EventLoop.defaultEventLoop();
        Server svr = new Server();
        svr.serve(new TetrisServer());
        try {
            svr.listen(server_port);
            System.out.println("Tetris RPC server is listening at " + server_port);
            loop.join();
        } catch (IOException exp1) {
            exp1.printStackTrace();
        } catch (InterruptedException exp2) {
            exp2.printStackTrace();
        }
    }
}
```

getBoardState()

```
public int [] getBoardState() {
    // Get board state
    int b_size = board.getBoardSize();
    Shape.Tetrominoe[] b_data = board.getBoardState();
    int [] state = new int[b_size];
    for (int i=0; i<b_size; i++) {
        state[i] = b_data[i].ordinal();
    }
    return state;
}
```

int [] step(int action_type)

```
public int [] step(int action_type)
{
    if (isGameOver())
        return getBoardState();
    switch(action_type)
    {
        case 0: // Move left
            move(-1);
            break;
        case 1: // Move right
            move(1);
            break;
        case 2: // Rotate left
            rotate(false);
            break;
        case 3: // Rotate right
            rotate(true);
            break;
        case 4: // Drop
            dropDown();
            break;
    }
    board.doGameCycle();
    return getBoardState();
}
```


MsgPack-RPC for Python

Install msgpack-rpc

- pip install msgpack-rpc-python

Random Play

```
import msgpackrpc;
import random
import time

server_ip = "localhost"
server_port = 10612

client = msgpackrpc.Client(msgpackrpc.Address(server_ip, server_port))

client.call('reset')

while True:
    action = random.randint(0, 4)
    state = client.call('step', action)
    #print(state)
    reward = client.call('getTotalReward')
    done = client.call('isDone')
    if done:
        print('Game over. Reward {}'.format(reward))
        client.call('reset')
```

Inheriting OpenAI Gym

```python
# Implement OpenAI Gym interface
import gym
import msgpackrpc # install msgpack-rpc-python
import numpy as np

class TetrisGym(gym.Env):
    board_width = 10
    board_height = 22

    def __init__(self, ip='localhost', port=10612):
        self.action_space = gym.spaces.Discrete(5)
        self.observation_space = gym.spaces.Box(
            low=0, high=7,
            shape=(self.board_height*self.board_width, 1),
            dtype=np.float32);

        self.client = msgpackrpc.Client(msgpackrpc.Address(ip, port))
        # initialize environment settings
        self.done = False

    def reset(self):
        return self.client.call('reset')

    def step(self, action):
        observation = self.client.call('step', action)
        reward = self.client.call('getTotalReward')
        self.done = self.client.call('isDone')
        info = None
        return observation, reward, self.done, info;
```

References

- https://msgpack.org/index.html