Lección 02

```
/**
 * 4. Publicar y subscribir objetos/interacciones (NUEVO)
 */
private void publishAndSubscribe() throws RTIexception {

        chat.interactionHandle = rtiamb
                        .getInteractionClassHandle("InteractionRoot.Chat");
        chat.parameterHandle = rtiamb.getParameterHandle("message",
                        chat.interactionHandle);

        rtiamb.publishInteractionClass(chat.interactionHandle);
        rtiamb.subscribeInteractionClass(chat.interactionHandle);
        logger.info("Publicar y subscribir");
}

/**
 * 6.1 Enviar una interacción (NUEVO)
 */
private void sendInteraction(String text) throws RTIexception {

        SuppliedParameters attributes = RtiFactoryFactory.getRtiFactory()
                        .createSuppliedParameters();
        byte[] textValue = EncodingHelpers.encodeString(text);
        attributes.add(chat.parameterHandle, textValue);

        byte[] tag = EncodingHelpers.encodeString(federateName);
        rtiamb.sendInteraction(chat.interactionHandle, attributes, tag);
}
```

Lección 02

```
/**
 * 5. Publicar y subscribir objetos/interacciones (NUEVO)
 */
private void publishAndSubscribe() throws RTIexception {

        classHandle = rtiamb.getObjectClassHandle("ObjectRoot.Punto");

        AttributeHandleSet attributes = RtiFactoryFactory.getRtiFactory()
                        .createAttributeHandleSet();
        String[] attributesNames = { "x", "y" };
        for (String name : attributesNames) {
                int attributeHandle = rtiamb.getAttributeHandle(name, classHandle);
                attributes.add(attributeHandle);
        }

        rtiamb.publishObjectClass(classHandle, attributes);
        rtiamb.subscribeObjectClassAttributes(classHandle, attributes);
        logger.info("Publicar y subscribir");
}

/**
 * 6. Registrar la instancia del objeto a actualizar (NUEVO)
 */
private int registerObjectInstance() throws RTIexception {
        logger.info("Registrar instancia del objeto (ObjectRoot.Punto)");
        return rtiamb.registerObjectInstance(classHandle);
}

/**
 * 7.1 Enviar actualización de los atributos (NUEVO)
 */
private void updateAttributeValues(int instanceHandle, int x, int y)
                throws RTIexception {
        SuppliedAttributes attributes = RtiFactoryFactory.getRtiFactory()
                        .createSuppliedAttributes();

        byte[] xValue = EncodingHelpers.encodeInt(x);
        byte[] yValue = EncodingHelpers.encodeInt(y);

        int xHandle = rtiamb.getAttributeHandle("x", classHandle);
        int yHandle = rtiamb.getAttributeHandle("y", classHandle);

        attributes.add(xHandle, xValue);
        attributes.add(yHandle, yValue);

        byte[] tag = EncodingHelpers.encodeString(federateName);
        rtiamb.updateAttributeValues(instanceHandle, attributes, tag);
}
```

```
/**
 * Lección 4: Integración con jME
 */
public class Leccion04 extends SimpleGame {

	private Node scene = new Node("Escena");
	private Node player = new Node("Jugador");

	private String federateName = "JoseM";
	private Leccion04_Federate federate;

	private boolean isUpdate = true;

	public static void main(String[] args) {
		Leccion04 app = new Leccion04();
		app.setConfigShowMode(ConfigShowMode.AlwaysShow, Leccion04.class
				.getClassLoader().getResource(
						"es/uji/taller/leccion04/logoEncuentro.png"));
		app.start();
	}

	@Override
	protected void simpleInitGame() {

		// Escena
		rootNode.attachChild(scene);
		buildTerrain();

		// Jugador
		rootNode.attachChild(player);
		buildPlayer();

		// KeyInputs
		buildInput();

		// Inicializar el federado
		federate = new Leccion04_Federate(this, federateName);
		federate.init();

		scene.updateGeometricState(0.0f, true);
		scene.updateRenderState();

		cam.setFrustumFar(5000);
		cam.setLocation(new Vector3f(5, 5, 20));
	}

	@Override
	protected void simpleUpdate() {

		if (KeyBindingManager.getKeyBindingManager().isValidCommand("avanza",
				true)) {
			player.getLocalTranslation().addLocal(0, 0, -0.01f);
```

```
                isUpdate = false;
        }
        if (KeyBindingManager.getKeyBindingManager().isValidCommand(
                        "retrocede", true)) {
                player.getLocalTranslation().addLocal(0, 0, 0.01f);
                isUpdate = false;
        }
        if (KeyBindingManager.getKeyBindingManager().isValidCommand("derecha",
                        true)) {
                player.getLocalTranslation().addLocal(0.01f, 0, 0);
                isUpdate = false;
        }
        if (KeyBindingManager.getKeyBindingManager().isValidCommand(
                        "izquierda", true)) {
                player.getLocalTranslation().addLocal(-0.01f, 0, 0);
                isUpdate = false;
        }

        if (!isUpdate){
                federate.send(player.getLocalTranslation().x,
                player.getLocalTranslation().z);
                isUpdate = true;
        } else {
                federate.advance();
        }
}

/**
 * ----------------------------------------------------------------------
 * Funciones auxiliares
 * ----------------------------------------------------------------------
 */

/**
 * Construye un bloque de terreno con desniveles
 */
private void buildTerrain() {
        Box floor = new Box("Suelo", Vector3f.ZERO, new Vector3f(10, 0.1f, 10));
        MaterialState ms = display.getRenderer().createMaterialState();
        ms.setAmbient(ColorRGBA.brown);
        floor.setRenderState(ms);
        floor.setLocalTranslation(new Vector3f(0, -0.1f, 0));
        floor.setModelBound(new BoundingBox());
        floor.updateModelBound();
        scene.attachChild(floor);
}

private void buildPlayer() {
        Box b = new Box("box", Vector3f.ZERO, new Vector3f(1, 1, 1));
        MaterialState ms = display.getRenderer().createMaterialState();
        ms.setAmbient(ColorRGBA.orange);
        b.setRenderState(ms);
```

```
            b.setModelBound(new BoundingBox());
            b.updateModelBound();
            player.attachChild(b);
        }

        private void buildInput() {
            KeyBindingManager.getKeyBindingManager().set("avanza", KeyInput.KEY_U);
            KeyBindingManager.getKeyBindingManager().set("retrocede",
                    KeyInput.KEY_J);
            KeyBindingManager.getKeyBindingManager().set("derecha", KeyInput.KEY_K);
            KeyBindingManager.getKeyBindingManager().set("izquierda",
                    KeyInput.KEY_H);
        }

        public void receive(float x, float y) {
            player.getLocalTranslation().set(new Vector3f(x,0,y));
        }
}
```