You can get a sample project here: https://github.com/Tichau/Schmup/tree/tp-factory if you don't want to use your project.

TP - Shoot'em up - Control Instantiation

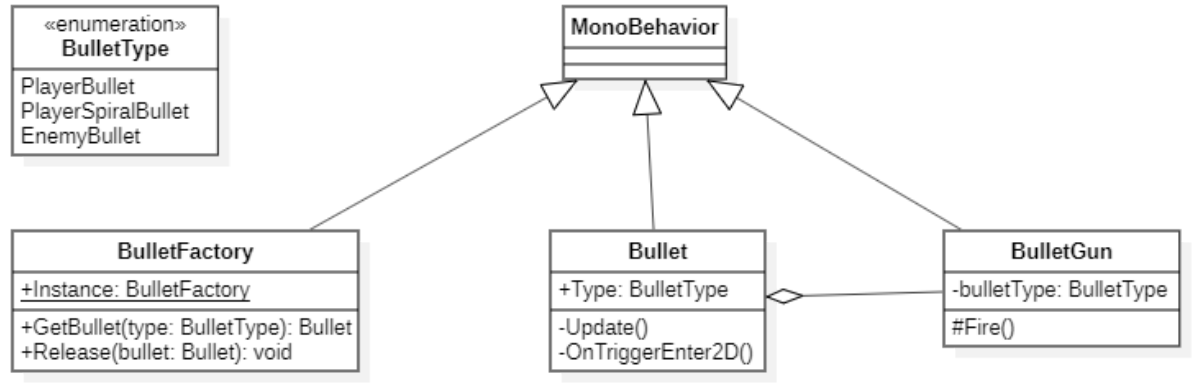
1. *The factory pattern*

In class-based programming, the factory method pattern is a creational pattern which uses factory methods to deal with the problem of creating objects without specifying the exact class of object that will be created. This is done by creating objects via calling a factory method rather than by calling a constructor.

Example

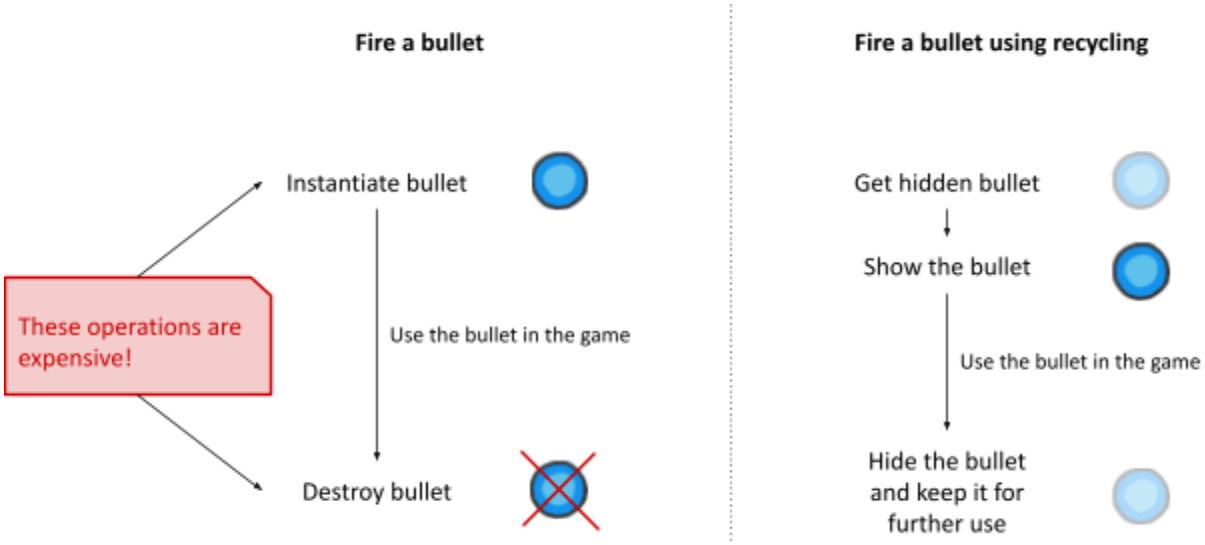
```
public interface IPeople
{
	string GetName();
}

public class Villagers : IPeople
{
	public string GetName()
	{
		return "Village Guy";
	}
}

public class CityPeople : IPeople
{
	public string GetName()
	{
		return "City Guy";
	}
}

public enum PeopleType
{
	RURAL,
	URBAN
}

/// <summary>
/// Implementation of Factory - Used to create objects
/// </summary>
public class Factory
{
	public IPeople GetPeople(PeopleType type)
	{
		IPeople people = null;
		switch (type)
		{
			case PeopleType.RURAL :
				people = new Villagers();
				break;
			case PeopleType.URBAN:
				people = new CityPeople();
				break;
			default:
				break;
		}
		return people;
	}
}
```

Question 1 : ***replace your existing bullet instantiation (in the fire() method of bulletgun) by a factory pattern (see UML diagram).***



Note: The factory should be a singleton (should be unique in your project) to handle all the bullet instances of the project.

2. *Pool recycling*



To avoid doing operations that cost us a lot of time, we will now recycle the bullets between uses. Instead of creating a bullet each time we need to fire, we will get one (previously used that we hide) and use it.

Question 2 : **Modify the bullet factory to recycle bullets instead of instantiate a new one each time.**

- **Add a list of available (unused) instances in the factory and initialize it at the beginning of the game.**
- **When a bullet is required, get an instance in this list (or create a new bullet if the list is empty).**
- **When a bullet is no longer useful (destroyed…), don't destroy it, deactivate it and put it back in the list.**
    - For better performance you can deactivate the rendering component instead of deactivating the entire gameobject (a lot faster). Don't forget to deactivate the gameplay component or you will have invisible dead bullets!

3. *Enemies instantiation*

Question 3 : **Implement the factory pattern for the enemies instantiation.**

Question 4 : **Implement the recycling of enemies (be careful with enemy types).**