Inheritance on classes: Parent class gives attributes to child class in MonoGame:

New Project --→ MonoGame --→ Windows --→Game --→ok

Right click --→ add --→new class --→call it (Sprite):

```
Texture2D texture;
Rectangle rectangle;
Public Sprite (Texture2D newTexture, Rectangle newRectangle)
{
      texture=newTexture;
      rectangle= newRectangle;
}

Public void Update()
{
}
Public void Draw(SpriteBatch spriteBatch)
{
spriteBatch.Draw(texture,rectangle, Color.White);

}
```

The class above provides the possibility to add any texture (sprite) inside the Game1.cs class without the need to define texture and rectangle for each individually.
Go back to Game1.cs and add Sprite class as a variable ball

```
      Sprite ball;
```

In loadContent:

```
      ball=new Sprite(Content.Load<Texture2D>("image"), new
Rectangle(10,10,100,100));
```

in Game1.cs in Draw:

```
spriteBatch.Begin();
ball.Draw(spriteBatch);
spriteBatch.End();
```

Completed Code:

class Sprite:

```
using Microsoft.Xna.Framework;
```

```
using Microsoft.Xna.Framework.Graphics;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Inheritance
{
   class Sprite
   {
      Texture2D texture;
      Rectangle rectangle;
      public Sprite (Texture2D newTexture, Rectangle newRectangle)
      {
         texture=newTexture;
         rectangle=newRectangle;
      }
      public void Update()
      {

      }
      public void Draw( SpriteBatch spriteBatch)
      {

         spriteBatch.Draw(texture,rectangle,Color.White);

      }
   }
}
```

class Game1:

```
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;

namespace Inheritance
{
   /// <summary>
   /// This is the main type for your game.
   /// </summary>
   public class Game1 : Game
   {
      GraphicsDeviceManager graphics;
      SpriteBatch spriteBatch;
      Sprite Fig;

      public Game1()
      {
         graphics = new GraphicsDeviceManager(this);
         Content.RootDirectory = "Content";
```

```
        }

        /// <summary>
        /// Allows the game to perform any initialization it needs to before starting to
run.
        /// This is where it can query for any required services and load any non-graphic
        /// related content.  Calling base.Initialize will enumerate through any components
        /// and initialize them as well.
        /// </summary>
        protected override void Initialize()
        {
            // TODO: Add your initialization logic here

            base.Initialize();
        }

        /// <summary>
        /// LoadContent will be called once per game and is the place to load
        /// all of your content.
        /// </summary>
        protected override void LoadContent()
        {
            // Create a new SpriteBatch, which can be used to draw textures.
            spriteBatch = new SpriteBatch(GraphicsDevice);
            Fig=new Sprite(Content.Load<Texture2D>("char1"),new Rectangle(3,3,100,200));

            // TODO: use this.Content to load your game content here
        }

        /// <summary>
        /// UnloadContent will be called once per game and is the place to unload
        /// game-specific content.
        /// </summary>
        protected override void UnloadContent()
        {
            // TODO: Unload any non ContentManager content here
        }

        /// <summary>
        /// Allows the game to run logic such as updating the world,
        /// checking for collisions, gathering input, and playing audio.
        /// </summary>
        /// <param name="gameTime">Provides a snapshot of timing values.</param>
        protected override void Update(GameTime gameTime)
        {
            if (GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed ||
Keyboard.GetState().IsKeyDown(Keys.Escape))
                Exit();

            // TODO: Add your update logic here

            base.Update(gameTime);
        }

        /// <summary>
        /// This is called when the game should draw itself.
        /// </summary>
        /// <param name="gameTime">Provides a snapshot of timing values.</param>
```

```
        protected override void Draw(GameTime gameTime)
        {
            GraphicsDevice.Clear(Color.CornflowerBlue);

            // TODO: Add your drawing code here
            spriteBatch.Begin();
            Fig.Draw(spriteBatch);
            spriteBatch.End();

            base.Draw(gameTime);
        }
    }
}
```

Add a child class in the Sprite.cs

All the attributes we want the child to take from parent needs to be defined in parent class and individual attributes goes in child class:

```
class Character: Sprite
{
Public Character(Texture2D newtexture, Rectangle newrectangle)
{
      texture=newTexture;
      rectangle=newRectangle;


}
}
```

Remove the code above from parent's class

```
      Public Texture2D texture;
      Public Rectangle rectangle;
```

In Sprite class: change the Update class to virtual otherwise it is going to be updated when we don't want it:

```
      Public virtual void Update(){}
```

In Character class define an update method

```
      Public override void Update()
      {
      If(Keyboard.GetState().IsKeyDown(Keys.Right))
      Rectangle.X+=3;
      }
```

Back in to Game1.cs

Change the following codes

```
      Character Fig;
      Fig=new Character(Content…..
```

In Update add:

```
Fig.Update();
```

Completed Code:

Sprite class:

```
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Graphics;

using Microsoft.Xna.Framework.Input;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Inheritance
{
   class Sprite
   {
      public Texture2D texture;
      public Rectangle rectangle;


      public virtual void Update()
      {

      }
      public void Draw( SpriteBatch spriteBatch)
      {

         spriteBatch.Draw(texture,rectangle,Color.White);

      }
   }
   // make this the child class of Sprite class
   class Character: Sprite
   {
      public Character (Texture2D newTexture, Rectangle newRectangle)
      {
         texture=newTexture;
         rectangle=newRectangle;



      }

      public override void Update()
      {
```

```
         if (Keyboard.GetState().IsKeyDown(Keys.Right))
            rectangle.X += 5; { if (rectangle.X + texture.Width > 800) rectangle.X = 800
- texture.Width; }
         if (Keyboard.GetState().IsKeyDown(Keys.Down))
            rectangle.Y += 5; { if (rectangle.Y + texture.Height > 500) rectangle.Y = 500
- texture.Height; }

         if (Keyboard.GetState().IsKeyDown(Keys.Left))
            rectangle.X -= 5; { if (rectangle.X <= 0) rectangle.X = 0; }

         if (Keyboard.GetState().IsKeyDown(Keys.Up))
            rectangle.Y -= 5; { if (rectangle.Y <= 0) rectangle.Y = 0; }




      }

   }
}
```

Game1 Class:

```
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;

namespace Inheritance
{
   /// <summary>
   /// This is the main type for your game.
   /// </summary>
   public class Game1 : Game
   {
      GraphicsDeviceManager graphics;
      SpriteBatch spriteBatch;
      Character Fig;

      public Game1()
      {
         graphics = new GraphicsDeviceManager(this);
         Content.RootDirectory = "Content";
      }

      /// <summary>
      /// Allows the game to perform any initialization it needs to before starting to
run.
      /// This is where it can query for any required services and load any non-graphic
      /// related content.  Calling base.Initialize will enumerate through any components
      /// and initialize them as well.
      /// </summary>
      protected override void Initialize()
      {
         // TODO: Add your initialization logic here

         base.Initialize();
```

```
        }

        /// <summary>
        /// LoadContent will be called once per game and is the place to load
        /// all of your content.
        /// </summary>
        protected override void LoadContent()
        {
            // Create a new SpriteBatch, which can be used to draw textures.
            spriteBatch = new SpriteBatch(GraphicsDevice);
            Fig = new Character(Content.Load<Texture2D>("char1"), new Rectangle(3, 3,
150,250));

            // TODO: use this.Content to load your game content here
        }

        /// <summary>
        /// UnloadContent will be called once per game and is the place to unload
        /// game-specific content.
        /// </summary>
        protected override void UnloadContent()
        {
            // TODO: Unload any non ContentManager content here
        }

        /// <summary>
        /// Allows the game to run logic such as updating the world,
        /// checking for collisions, gathering input, and playing audio.
        /// </summary>
        /// <param name="gameTime">Provides a snapshot of timing values.</param>
        protected override void Update(GameTime gameTime)
        {
            if (Keyboard.GetState().IsKeyDown(Keys.Escape))
                this.Exit();

            Fig.Update();

            base.Update(gameTime);
        }

        /// <summary>
        /// This is called when the game should draw itself.
        /// </summary>
        /// <param name="gameTime">Provides a snapshot of timing values.</param>
        protected override void Draw(GameTime gameTime)
        {
            GraphicsDevice.Clear(Color.CornflowerBlue);

            // TODO: Add your drawing code here
            spriteBatch.Begin();
            Fig.Draw(spriteBatch);
            spriteBatch.End();

            base.Draw(gameTime);
        }
    }
}
```